Hewlett-Packard -- Portable Computer Division

Research and Development Laboratory

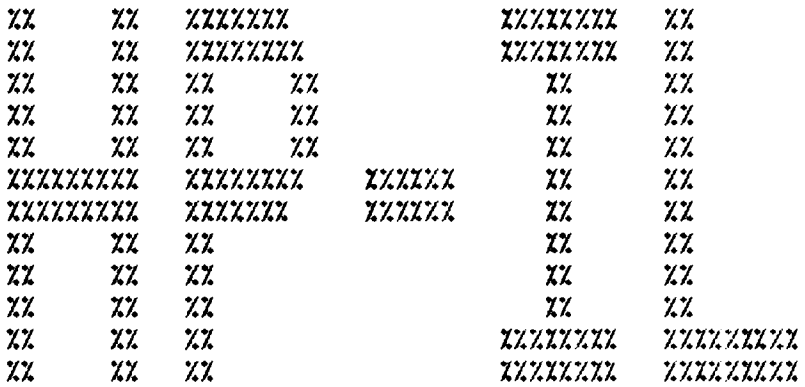
Corvallis, Oregon

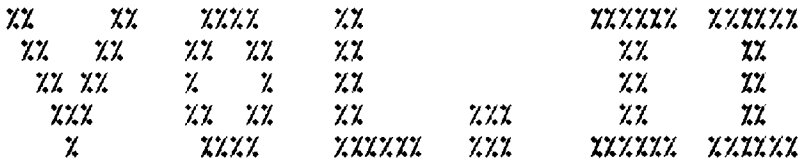
HP-71 HP-IL Module

Internal Design Specification

VOLUME II

Source Listings





Version 1.0 - Preliminary - January 1, 1984

Copyright (c) Hewlett-Packard Company 1984

**** 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
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 HP-IL Module. The program modules which comprise the 16K-byte ROM are presented here in address order according to their position in the ROM, from lowest address to highest address. For purposes of presentation the modules are assembled relative to a ROM starting address of F0000 hex. In actuality the ROM is soft-configurable, and may be automatically configured by the HP-71 to others sections of the address space.

The following sections give a list of the program 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 object (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 or handles that poll. In this preliminary version of this document, supported entry points are not yet indicated in the source listings as they are in the HP-71 operating system source listings. However, the poll interfaces and certain entry point interfaces will be supported.

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

LIST OF MODULES IN ADDRESS ORDER — CHAPTER 2

Address Range	Module	Title
NZ%RST	F0000 - F0007	ROM Start (Header)
NZ%TBL	F0008 - F0409	Lexical Analyzer Tables--ID=FF
NZ%ERR	F040A - F06B4	Error Message Table
NZ%DIR	F06B5 - F07C1	Directory Section
NZ%GPR	F07C2 - F0F99	General Routines
NZ%BAS	F0F9A - F1F33	BASIC Routines
SC%ENT	F1F34 - F2C95	ENTER Execution
NZ%UTL	F2C96 - F2ED6	User Utility Routines
NZ%BIF	F2ED7 - F362D	Basic interface
NZ%IOB	F362E - F3636	I/O Buffer Routines
NZ%DSP	F3637 - F3BF6	Display Driver
NZ%BUT	F3BF7 - F4292	BASIC Utilities
NZ%CAS	F4293 - F511A	Cassette Routines
NZ%HND	F511B - F5E90	Poll Handlers
NZ%CAT	F5E91 - F66D1	HP-IL CAT
NZ%IOR	F66D2 - F6BD7	I/O (NEW Mailbox)
NZ%FRA	F6BD8 - F6D55	HP-IL Frame Routines
NZ%LOW	F6D56 - F6E18	Low-level User HP-IL
NZ%FXQ	F6E19 - F74FC	File Execution
NZ%PAR	F74FD - F7BD2	HP-IL Parse Routines
NZ%DEC	F7BD3 - F7EF0	HP-IL Decompile Routines
JP%ZER	F7FFC - F7FFD	Zero File - ROM Checksum
JP%ZER	F7FFE - F7FFF	Zero File - End of chain
NZ%SYM	No Address	Symbolic Assignments

LIST OF MODULES SORTED BY MODULE NAME — CHAPTER 3

Module	Address Range	Title
JP%ZER	F7FFC - F7FFD	Zero File - ROM Checksum
JP%ZER	F7FFE - F7FFF	Zero File - End of chain
NZ%BAS	F0F9A - F1F33	BASIC Routines
NZ%BIF	F2ED7 - F362D	Basic interface
NZ%BUT	F3BF7 - F4292	BASIC Utilities
NZ%CAS	F4293 - F511A	Cassette Routines
NZ%CAT	F5E91 - F66D1	HP-IL CAT
NZ%DEC	F7BD3 - F7EF0	HP-IL Decompile Routines
NZ%DIR	F06B5 - F07C1	Directory Section
NZ%DSP	F3637 - F3BF6	Display Driver
NZ%ERR	F040A - F06B4	Error Message Table
NZ%FRA	F6BD8 - F6D55	HP-IL Frame Routines
NZ%FXQ	F6E19 - F74FC	File Execution
NZ%GPR	F07C2 - F0F99	General Routines
NZ%HND	F511B - F5E90	Poll Handlers
NZ%IOB	F362E - F3636	I/O Buffer Routines
NZ%IOR	F66D2 - F6BD7	I/O (NEW Mailbox)
NZ%LOW	F6D56 - F6E18	Low-level User HP-IL
NZ%PAR	F74FD - F7BD2	HP-IL Parse Routines
NZ%RST	F0000 - F0007	ROM Start (Header)
NZ%SYM	No Address	Symbolic Assignments
NZ%TBL	F0008 - F0409	Lexical Analyzer Tables--ID=FF
NZ%UTL	F2C96 - F2ED6	User Utility Routines
SC%ENT	F1F34 - F2C95	ENTER Execution

```
          ************************************************
          ************************************************
          **                                            **
          ** H   H  PPPP   III  L          1    BBBB    **
          ** H   H  P   P   I   L     ::: 11    B   B   **
          ** H   H  P   P   I   L     :::  1    B   B   **
          ** HHHHH  PPPP    I   L          1    BBBB    **
          ** H   H  P       I   L     :::  1    B   B   **
          ** H   H  P       I   L     :::  1    B   B   **
          ** H   H  P      III  LLLLL    111   BBBB    **
          **                                            **
          ************************************************
          ************************************************
/SLOAD:  Duplicate entry point A-MULT found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CATC++ found in modules NZ%PAR and TI%R6S
/SLOAD:  Duplicate entry point CONVUC found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSLC1  found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSLC10 found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSLC11 found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSLC12 found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSLC13 found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSLC14 found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSLC15 found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSLC2  found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSLC3  found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSLC4  found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSLC5  found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSLC6  found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSLC7  found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSLC8  found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSLC9  found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSRC1  found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSRC10 found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSRC11 found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSRC12 found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSRC13 found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSRC14 found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSRC15 found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSRC2  found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSRC3  found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSRC4  found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSRC5  found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSRC6  found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSRC7  found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSRC8  found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point CSRC9  found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point D1=AVE found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point D1@AVS found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point EXPEX+ found in modules NZ%BUT and TI%R6S
/SLOAD:  Duplicate entry point FIND   found in modules NZ%BAS and TI%R6S
/SLOAD:  Duplicate entry point FINDF+ found in modules NZ%CAS and TI%R6S
/SLOAD:  Duplicate entry point GETST- found in modules NZ%IOR and TI%R6S
/SLOAD:  Duplicate entry point NUMCK  found in modules NZ%PAR and TI%R6S
/SLOAD:  Duplicate entry point OUT3TC found in modules NZ%PAR and TI%R6S
/SLOAD:  Duplicate entry point OUTBYT found in modules NZ%PAR and TI%R6S
/SLOAD:  Duplicate entry point OUTNBC found in modules NZ%PAR and TI%R6S
/SLOAD:  Duplicate entry point POP1N  found in modules NZ%LOW and TI%R6S
/SLOAD:  Duplicate entry point RANGE  found in modules NZ%GPR and TI%R6S
/SLOAD:  Duplicate entry point RDINFO found in modules NZ%BUT and TI%R6S
/SLOAD:  Duplicate entry point READIN found in modules NZ%BAS and TI%R6S
```

```
/SLOAD:  Duplicate entry point RESPTR found in modules NZ%PAR and TI%R6S
/SLOAD:  Duplicate entry point SENDIT found in modules NZ%IOR and TI%R6S
SLOAD Rev. 2309/Ver. 1.40

Output module:
TI%HP7:TI:MS::-1    Start=F0000 End=F7FFF Length=08000 Syms=2489 Refs=1605
     Date=Tue Jan 24, 1984   5:40 pm  Title=(TI%R6S) HPIL Interface ROM

Source modules:
NZ%RST::MS          Start=F0000 End=F0007 Length=00008
     Date=Tue Jan 17, 1984  12:21 pm  Title=Rom start (header) <830927.1416>

NZ%TBL::MS          Start=F0008 End=F0409 Length=00402
     Date=Tue Jan 24, 1984   5:39 pm  Title=Lexical Analyzer Tables--ID=FF

NZ%ERR::MS          Start=F040A End=F06B4 Length=002AB
     Date=Tue Jan 17, 1984  12:05 pm  Title=

NZ%DIR::MS          Start=F06B5 End=F07C1 Length=0010D
     Date=Tue Jan 17, 1984  12:03 pm  Title=DIRECTORY SECTION <840106.1804>

NZ%GPR::MS          Start=F07C2 End=F0F99 Length=007D8
     Date=Tue Jan 17, 1984  12:08 pm  Title=GENERAL ROUTINES <840106.1701>

NZ%BAS::MS          Start=F0F9A End=F1F33 Length=00F9A
     Date=Tue Jan 17, 1984  11:42 am  Title=BASIC ROUTINES <840116.1657>

SC%ENT::MS          Start=F1F34 End=F2C95 Length=00D62
     Date=Tue Jan 17, 1984   1:23 pm  Title=ENTER Execution <840113.1057>

NZ%UTL::MS          Start=F2C96 End=F2ED6 Length=00241
     Date=Tue Jan 17, 1984  12:22 pm  Title=User Utility Routines <830927.1255>

NZ%BIF::MS          Start=F2ED7 End=F362D Length=00757
     Date=Tue Jan 24, 1984   5:33 pm  Title=Basic interface <840124.1345>

NZ%IOB::MS          Start=F362E End=F3636 Length=00009
     Date=Tue Jan 17, 1984  12:16 pm  Title=I/O Buffer routines <830927.1450>

NZ%DSP::MS          Start=F3637 End=F3BF6 Length=005C0
     Date=Tue Jan 17, 1984  12:03 pm  Title=Display driver <831108.0941>

NZ%BUT::MS          Start=F3BF7 End=F4292 Length=0069C
     Date=Tue Jan 17, 1984  11:52 am  Title=BASIC UTILITIES <840104.1515>

NZ%CAS::MS          Start=F4293 End=F511A Length=00E88
     Date=Tue Jan 17, 1984  11:55 am  Title=CASSETTE ROUTINES<831221.1632>

NZ%HND::MS          Start=F511B End=F5E90 Length=00D76
     Date=Tue Jan 17, 1984  12:12 pm  Title=POLL HANDLERS <840106.0805>

NZ%CAT::MS          Start=F5E91 End=F66D1 Length=00841
     Date=Tue Jan 17, 1984  11:59 am  Title=HPIL CAT <840106.1936>

NZ%IOR::MS          Start=F66D2 End=F6BD7 Length=00506
     Date=Tue Jan 17, 1984  12:16 pm  Title=I/O(NEW Mailbox)<831101.2117>

NZ%FRA::MS          Start=F6BD8 End=F6D55 Length=0017E
     Date=Tue Jan 17, 1984  12:06 pm  Title=PIL Frame Routines<831012.1534>
```

```
NZ%LOW::MS          Start=F6D56 End=F6E18 Length=00C3
      Date=Tue Jan 17, 1984  12:17 pm  Title=Low-level USER HP-IL <830927.1414>

NZ%FXQ::MS          Start=F6E19 End=F74FC Length=06E4
      Date=Tue Jan 17, 1984  12:06 pm  Title=File Execution <840113.1351>

NZ%PAR::MS          Start=F74FD End=F7BD2 Length=06D6
      Date=Tue Jan 17, 1984  12:18 pm  Title=NZ'S PARSE ROUTINES <831128.2333>

NZ%DEC::MS          Start=F7BD3 End=F7EF0 Length=031E
      Date=Tue Jan 17, 1984  12:02 pm  Title=PIL DECOMPILE ROUTINES<831027.1220>

NZ%SYM::MS          Module Contains No Code
      Date=Tue Jan 17, 1984  12:21 pm  Title=Symbolic Assignments <831220.1633>

SA%RMT              Start=F7FFC End=F7FFD Length=00002
      Date=Mon Nov 22, 1982   8:48 am  Title=ROM/IRAM tail end

SA%RMT              Start=F7FFE End=F7FFF Length=00002
      Date=Mon Nov 22, 1982   8:48 am  Title=ROM/IRAM tail end

TI%R6S              Module Contains No Code
      Date=Tue Jan 17, 1984  10:07 am  Title=Titan External Symbol File
```

Saturn Long Cross Reference Listing

```
#CK     =  03356 TI%R6S         -
#Timeo  =  0001E NZ%SYM         - F16DE NZ%BAS(00744) Type=0.0 Nibs=2

-LINE   =  15275 TI%R6S         -

1/X15   =  0C33E TI%R6S         -

?A=CLN  =  F7EE6 NZ%DEC         -
?A=CM+  =  F7EDB NZ%DEC         - F75A6 NZ%PAR(000A9) Type=1.1 Nibs=4 Dist=00935
                                + F76E0 NZ%PAR(001E3) Type=1.1 Nibs=3 Dist=007FB
?A=CMA  =  F7ED8 NZ%DEC         -
?PRFI+  =  17380 TI%R6S         -
?PRFIL  =  1737E TI%R6S         -

A-MULT  =  F0E22 NZ%GPR         - F5276 NZ%HND(0015B) Type=1.1 Nibs=4 Dist=04454
                                + F650A NZ%CAT(00679) Type=1.1 Nibs=4 Dist=056E8
ACCEPT  =  0450F TI%R6S         -
ACOS12  =  0DBD3 TI%R6S         -
ACOS15  =  0DBD7 TI%R6S         -
ACTIVE  =  2F5A8 TI%R6S         -
AD15M   =  0C366 TI%R6S         -
AD15S   =  0E19D TI%R6S         -
AD15s   =  0C369 TI%R6S         -
AD2-12  =  0C35F TI%R6S         -
AD2-15  =  0C363 TI%R6S         -
ADDF    =  0C372 TI%R6S         -
ADDONE  =  0C330 TI%R6S         -
ADDP    =  03A03 TI%R6S         -
ADDRSS  =  0F527 TI%R6S         -
ADHEAD  =  181B7 TI%R6S         -
ADJA    =  1289A TI%R6S         -
ADJN    =  12825 TI%R6S         -
ADRS40  =  0F52B TI%R6S         -
ADRS50  =  0F551 TI%R6S         -
ADRS80  =  0F567 TI%R6S         -
ADRSUB  =  0F4CF TI%R6S         -
ALLDUN  =  04BEF TI%R6S         -
ALMSRV  =  1257D TI%R6S         -
ALRM1   =  2F719 TI%R6S         -
ALRM2   =  2F725 TI%R6S         -
ALRM3   =  2F731 TI%R6S         -
ALRM4   =  2F73D TI%R6S         -
ALRM5   =  2F749 TI%R6S         -
ALRM6   =  2F755 TI%R6S         -
ALRNOG  =  F0EA8 NZ%GPR         -
ALRNOS  =  F0EDA NZ%GPR         -
ANN1.5  =  2E101 TI%R6S         -
ANNAD1  =  2E100 TI%R6S         -
ANNAD2  =  2E102 TI%R6S         -
ANNAD3  =  2E34C TI%R6S         -
ANNAD4  =  2E34E TI%R6S         -
ARG12   =  0D67B TI%R6S         -
ARG15   =  0D67F TI%R6S         -
ARGERR  =  0BF19 TI%R6S         -
ARGF    =  0D6A4 TI%R6S         -
ARGPR+  =  0E8EB TI%R6S         -
ARGPRP  =  0E8EF TI%R6S         -
ARGST-  =  0E910 TI%R6S         -
```

```
ARGSTA =  0E90C TI%R6S        -
ARITH  =  061E0 TI%R6S        -
ARLNOS =  F0EC2 NZ%GPR        -
ARRYCK =  0366A TI%R6S        -
ARYDC  =  05178 TI%R6S        -
ARYELM =  0B5A7 TI%R6S        -
ARYSIZ =  0B61B TI%R6S        -
ASCICK =  0514E TI%R6S        -
ASCII  =  0079B TI%R6S        -
ASGNIO =  F19CD NZ%BAS        - F0116 NZ%TBL(0010E) Type=1.2 Nibs=5 Dist=018B7
ASGNd  =  F7D06 NZ%DEC        - F19C3 NZ%BAS(00A29) Type=1.2 Nibs=5 Dist=06343
ASGNp  =  F769C NZ%PAR        - F19C8 NZ%BAS(00A2E) Type=1.2 Nibs=5 Dist=05CD4
ASIN12 =  0DBC8 TI%R6S        -
ASIN15 =  0DBCC TI%R6S        -
ASLC1  =  F0F11 NZ%GPR        -
ASLC10 =  F0F19 NZ%GPR        -
ASLC11 =  F0F1C NZ%GPR        -
ASLC12 =  F0F1F NZ%GPR        - F576C NZ%HND(00651) Type=1.1 Nibs=4 Dist=0484D
                              + F6F34 NZ%FXQ(0011B) Type=1.1 Nibs=4 Dist=06015
ASLC13 =  F0F22 NZ%GPR        -
ASLC14 =  F0F25 NZ%GPR        -
ASLC15 =  F0F28 NZ%GPR        -
ASLC2  =  F0F0E NZ%GPR        - F1AEB NZ%BAS(00B51) Type=1.1 Nibs=4 Dist=00BDD
ASLC3  =  F0F0B NZ%GPR        - F49CA NZ%CAS(00737) Type=1.1 Nibs=4 Dist=03ABF
                              + F5248 NZ%HND(0012D) Type=1.1 Nibs=4 Dist=0433D
ASLC4  =  F0F08 NZ%GPR        - F1421 NZ%BAS(00487) Type=1.1 Nibs=3 Dist=00519
                              + F35CA NZ%BIF(006F3) Type=1.1 Nibs=4 Dist=026C2
                              + F43BD NZ%CAS(0012A) Type=1.1 Nibs=4 Dist=034B5
                              + F43CF NZ%CAS(0013C) Type=1.1 Nibs=4 Dist=034C7
                              + F453C NZ%CAS(002A9) Type=1.1 Nibs=4 Dist=03634
                              + F56AC NZ%HND(00591) Type=1.1 Nibs=4 Dist=047A4
ASLC5  =  F0F05 NZ%GPR        - F18A5 NZ%BAS(0090B) Type=1.1 Nibs=4 Dist=009A0
ASLC6  =  F0F02 NZ%GPR        - F56CC NZ%HND(005B1) Type=1.1 Nibs=4 Dist=047CA
ASLC7  =  F0EFF NZ%GPR        -
ASLC8  =  F0EFC NZ%GPR        -
ASLC9  =  F0F16 NZ%GPR        - F4936 NZ%CAS(006A3) Type=1.1 Nibs=4 Dist=03A20
ASLW3  =  0ED21 TI%R6S        -
ASLW4  =  0ED1E TI%R6S        -
ASLW5  =  0ED1B TI%R6S        -
ASNMNT =  0F5E0 TI%R6S        -
ASRC1  =  F0F28 NZ%GPR        -
ASRC10 =  F0F02 NZ%GPR        - F4711 NZ%CAS(0047E) Type=1.1 Nibs=4 Dist=0380F
                              + F56E4 NZ%HND(005C9) Type=1.1 Nibs=4 Dist=047E2
ASRC11 =  F0F05 NZ%GPR        -
ASRC12 =  F0F08 NZ%GPR        -
ASRC13 =  F0F0B NZ%GPR        -
ASRC14 =  F0F0E NZ%GPR        -
ASRC15 =  F0F11 NZ%GPR        -
ASRC2  =  F0F25 NZ%GPR        - F1AC7 NZ%BAS(00B2D) Type=1.1 Nibs=4 Dist=00BA2
ASRC3  =  F0F22 NZ%GPR        - F49BF NZ%CAS(0072C) Type=1.1 Nibs=4 Dist=03A9D
                              + F55E3 NZ%HND(004C8) Type=1.1 Nibs=4 Dist=046C1
ASRC4  =  F0F1F NZ%GPR        - F137B NZ%BAS(003E1) Type=1.1 Nibs=3 Dist=0045C
                              + F140C NZ%BAS(00472) Type=1.1 Nibs=3 Dist=004ED
                              + F14CB NZ%BAS(00531) Type=1.1 Nibs=3 Dist=005AC
                              + F4AA6 NZ%CAS(00813) Type=1.0 Nibs=4 Dist=03B87
                              + F5574 NZ%HND(00459) Type=1.1 Nibs=4 Dist=04655
                              + F6EAF NZ%FXQ(00096) Type=1.1 Nibs=4 Dist=05F90
ASRC5  =  F0F1C NZ%GPR        - F18B3 NZ%BAS(00919) Type=1.1 Nibs=4 Dist=00997
                              + F2B9D SC%ENT(00C69) Type=1.1 Nibs=4 Dist=01C81
                              + F415E NZ%BUT(00567) Type=1.1 Nibs=4 Dist=03242
```

```
                                + F467E NZ%CAS(003EB) Type=1.1 Nibs=4 Dist=03762
                                + F5593 NZ%HND(00478) Type=1.1 Nibs=4 Dist=04677
                                + F5A1A NZ%HND(008FF) Type=1.1 Nibs=4 Dist=04AFE
ASRC6  =  F0F19 NZ%GPR          -
ASRC7  =  F0F16 NZ%GPR          -
ASRC8  =  F0EFC NZ%GPR          - F4E72 NZ%CAS(00BDF) Type=1.1 Nibs=4 Dist=03F76
ASRC9  =  F0EFF NZ%GPR          - F49A6 NZ%CAS(00713) Type=1.1 Nibs=4 Dist=03AA7
ASRW3  =  0ED10 TI%R6S          -
ASRW4  =  0ED0D TI%R6S          -
ASRW5  =  0ED0A TI%R6S          -
ATAN15 =  0DBBE TI%R6S          -
ATNCHK =  F0BC5 NZ%GPR          - F34FF NZ%BIF(00628) Type=1.1 Nibs=4 Dist=0293A
ATNCLR =  00510 TI%R6S          -
ATNDIS =  2F441 TI%R6S          -
ATNFLG =  2F442 TI%R6S          - F0BD1 NZ%GPR(0040F) Type=0.0 Nibs=5
                                + F2B43 SC%ENT(00C0F) Type=0.0 Nibs=5
                                + F2F81 NZ%BIF(000AA) Type=0.0 Nibs=5
                                + F67A4 NZ%IOR(000D2) Type=0.0 Nibs=5
                                + F6A0D NZ%IOR(0033B) Type=0.0 Nibs=5
                                + F6AE2 NZ%IOR(00410) Type=0.0 Nibs=5
AUTINC =  2F6CB TI%R6S          -
AVE=C  =  18BBB TI%R6S          -
AVE=D1 =  18BB8 TI%R6S          - F21BD SC%ENT(00289) Type=0.1 Nibs=5
AVM+16 =  F40C2 NZ%BUT          -
AVMEME =  2F599 TI%R6S          - F0F76 NZ%GPR(007B4) Type=0.0 Nibs=5
AVMEMS =  2F594 TI%R6S          - F0F7F NZ%GPR(007BD) Type=0.0 Nibs=5
AVS2DS =  09708 TI%R6S          -
Attn   =  0000C NZ%SYM          - F0BC7 NZ%GPR(00405) Type=0.0 Nibs=1
                                + F679C NZ%IOR(000CA) Type=0.0 Nibs=1
                                + F69B1 NZ%IOR(002DF) Type=0.0 Nibs=1
                                + F69F5 NZ%IOR(00323) Type=0.0 Nibs=1
                                + F6A26 NZ%IOR(00354) Type=0.0 Nibs=1
                                + F6A88 NZ%IOR(003B6) Type=0.0 Nibs=1
                                + F6AAA NZ%IOR(003D8) Type=0.0 Nibs=1
                                + F6B62 NZ%IOR(00490) Type=0.0 Nibs=1
                                + F6B93 NZ%IOR(004C1) Type=0.0 Nibs=1

BACK   =  1BA4F TI%R6S          -
BACK1B =  13B0C TI%R6S          -
BACK2B =  13B0A TI%R6S          -
BACK3B =  13B08 TI%R6S          -
BAKCHR =  F3FC2 NZ%BUT          - F1ADD NZ%BAS(00B43) Type=1.1 Nibs=4 Dist=024E5
                                + F74D0 NZ%FXQ(006B7) Type=1.0 Nibs=4 Dist=0350E
BASCHA =  07741 TI%R6S          -
BASCHK =  0773E TI%R6S          -
BASE   =  0F953 TI%R6S          -
BASICs =  000B5 TI%R6S          -
BDISPJ =  F3637 NZ%DSP          - F3001 NZ%BIF(0012A) Type=1.2 Nibs=5 Dist=00636
BEEP   =  0EA6E TI%R6S          -
BF2DSP =  01C0E TI%R6S          - F1871 NZ%BAS(008D7) Type=0.1 Nibs=5
                                + F1914 NZ%BAS(0097A) Type=0.1 Nibs=5
                                + F5F35 NZ%CAT(000A4) Type=0.1 Nibs=5
BF2STK =  18663 TI%R6S          -
BIASA+ =  0D52D TI%R6S          -
BIASC+ =  0D540 TI%R6S          -
BIG    =  0B747 TI%R6S          -
BINAND =  F1E66 NZ%BAS          - F0098 NZ%TBL(00090) Type=1.2 Nibs=5 Dist=01DCE
BINCMP =  F1EB7 NZ%BAS          - F00A1 NZ%TBL(00099) Type=1.2 Nibs=5 Dist=01E16
BINEOR =  F1E96 NZ%BAS          - F00AA NZ%TBL(000A2) Type=1.2 Nibs=5 Dist=01DEC
BINIOR =  F1E86 NZ%BAS          - F00B3 NZ%TBL(000AB) Type=1.2 Nibs=5 Dist=01DD3
```

```
BIT    =  F1ECF NZ%BAS        - F00BC NZ%TBL(000B4) Type=1.2 Nibs=5 Dist=01E13
BLANK  =  F7B2A NZ%PAR        -
BLANKC =  F0F5E NZ%GPR        - F1811 NZ%BAS(00877) Type=1.1 Nibs=4 Dist=008B3
                              + F38C6 NZ%DSP(0028F) Type=1.1 Nibs=4 Dist=02968
                              + F4429 NZ%CAS(00196) Type=1.1 Nibs=4 Dist=034CB
                              + F58DE NZ%HND(007C3) Type=1.1 Nibs=4 Dist=04980
                              + F63C3 NZ%CAT(00532) Type=1.1 Nibs=4 Dist=05465
                              + F7433 NZ%FXQ(0061A) Type=1.1 Nibs=4 Dist=064D5
BLDBIT =  019BC TI%R6S        -
BLDCAT =  F6395 NZ%CAT        -
BLDCON =  16279 TI%R6S        - F213E SC%ENT(0020A) Type=0.1 Nibs=5
BLDDSP =  01898 TI%R6S        -
BLDLCD =  0189C TI%R6S        -
BLNKCK =  051C1 TI%R6S        -
BOPNM- =  1B864 TI%R6S        -
BP+C   =  0EB40 TI%R6S        -
BRT30  =  0DBE3 TI%R6S        -
BRTF   =  0DC15 TI%R6S        -
BSCEX2 =  0743A TI%R6S        -
BSCEXC =  07437 TI%R6S        -
BSCEXT =  075CF TI%R6S        -
BSERR  =  0939A TI%R6S        - F1A32 NZ%BAS(00A98) Type=0.1 Nibs=5
BitsOK =  00001 TI%R6S        -
BldIM+ =  1BA6A TI%R6S        -
BldIMA =  1BA66 TI%R6S        -
BldIMG =  1BA68 TI%R6S        -

C+A2D1 =  1C053 TI%R6S        -
CALBIN =  18D8C TI%R6S        -
CALL   =  18DAE TI%R6S        -
CALLP  =  0389C TI%R6S        -
CALSTK =  2F5AD TI%R6S        -
CAT$20 =  06746 TI%R6S        -
CATC++ =  F7B11 NZ%PAR        -
CATCH+ =  03F69 TI%R6S        - F7B16 NZ%PAR(00619) Type=0.1 Nibs=5
CATCHR =  03F70 TI%R6S        -
CATEDT =  06435 TI%R6S        -
CHAIN+ =  07C12 TI%R6S        -
CHAIN- =  07C1C TI%R6S        - F56FA NZ%HND(005DF) Type=0.1 Nibs=5
CHECKD =  F6864 NZ%IOR        -
CHEDIT =  14C99 TI%R6S        -
CHIRP  =  0EC5A TI%R6S        -
CHKAIO =  F411B NZ%BUT        - F7217 NZ%FXQ(003FE) Type=1.1 Nibs=4 Dist=030FC
CHKASN =  F3CEC NZ%BUT        - F0FBC NZ%BAS(00022) Type=1.1 Nibs=4 Dist=02D30
                              + F22AE SC%ENT(0037A) Type=1.1 Nibs=4 Dist=01A3E
                              + F2FEA NZ%BIF(00113) Type=1.1 Nibs=4 Dist=00D02
                              + F3643 NZ%DSP(0000C) Type=1.1 Nibs=3 Dist=006A9
                              + F53ED NZ%HND(002D2) Type=1.1 Nibs=4 Dist=01701
CHKBIT =  F430E NZ%CAS        - F5601 NZ%HND(004E6) Type=1.1 Nibs=4 Dist=012F3
                              + F5786 NZ%HND(0066B) Type=1.1 Nibs=4 Dist=01478
CHKEND =  F6881 NZ%IOR        -
CHKEOL =  13D6D TI%R6S        - F1FC5 SC%ENT(00091) Type=0.1 Nibs=5
                              + F2233 SC%ENT(002FF) Type=0.1 Nibs=5
CHKMAS =  F42F1 NZ%CAS        - F11F2 NZ%BAS(00258) Type=1.1 Nibs=4 Dist=030FF
                              + F14F3 NZ%BAS(00559) Type=1.1 Nibs=4 Dist=02DFE
                              + F3596 NZ%BIF(006BF) Type=1.1 Nibs=4 Dist=00D5B
                              + F51C8 NZ%HND(000AD) Type=1.1 Nibs=4 Dist=00ED7
                              + F60E8 NZ%CAT(00257) Type=1.1 Nibs=4 Dist=01DF7
CHKSEC =  F5CEB NZ%HND        - F4BD6 NZ%CAS(00943) Type=1.1 Nibs=4 Dist=01115
CHKSET =  F31DE NZ%BIF        - F0C31 NZ%GPR(0046F) Type=1.1 Nibs=4 Dist=025AD
```

```
CHKST+ =  F31F5 NZ%BIF        - F6E06 NZ%LOW(000B0) Type=1.1 Nibs=4 Dist=03C11
CHKSTS =  F0C24 NZ%GPR        - F2AA1 SC%ENT(00B6D) Type=1.1 Nibs=4 Dist=01E7D
                              + F3069 NZ%BIF(00192) Type=1.1 Nibs=4 Dist=02445
CHKmem =  012C7 TI%R6S        -
CHN#SV =  2F96F TI%R6S        - F2219 SC%ENT(002E5) Type=0.0 Nibs=5
CHNHED =  0F579 TI%R6S        -
CHNLST =  2F5BE TI%R6S        -
CK"ON" =  076AD TI%R6S        -
CK=ATN =  F6A03 NZ%IOR        -
CK=ATn =  F6A08 NZ%IOR        - F5F66 NZ%CAT(000D5) Type=1.1 Nibs=4 Dist=00AA2
CKBITL =  F5784 NZ%HND        - F5E92 NZ%CAT(00001) Type=1.1 Nibs=3 Dist=0070E
CKHPI+ =  F5790 NZ%HND        -
CKHPIL =  F578D NZ%HND        -
CKINF- =  18534 TI%R6S        - F661B NZ%CAT(0078A) Type=0.1 Nibs=5
CKINF0 =  18542 TI%R6S        -
CKLOP# =  F297B SC%ENT        - F153E NZ%BAS(005A4) Type=1.1 Nibs=4 Dist=0143D
                              + F1981 NZ%BAS(009E7) Type=1.1 Nibs=4 Dist=00FFA
CKSREQ =  00721 TI%R6S        -
CKSTR  =  F7A84 NZ%PAR        -
CKSUM2 =  0AA81 TI%R6S        -
CKSUM3 =  153A9 TI%R6S        -
CKSUM4 =  1DBA6 TI%R6S        -
CKmode =  F28FF SC%ENT        - F15CA NZ%BAS(00630) Type=1.1 Nibs=4 Dist=01335
CLASSA =  0D590 TI%R6S        -
CLCBFR =  2F576 TI%R6S        -
CLCSTK =  2F585 TI%R6S        -
CLEAR  =  F1585 NZ%BAS        - F010D NZ%TBL(00105) Type=1.2 Nibs=5 Dist=01478
CLEARN =  F4318 NZ%CAS        -
CLEARd =  F7CC7 NZ%DEC        - F157B NZ%BAS(005E1) Type=1.2 Nibs=5 Dist=0674C
CLEARp =  F761E NZ%PAR        - F1580 NZ%BAS(005E6) Type=1.2 Nibs=5 Dist=0609E
CLLOOP =  F431D NZ%CAS        -
CLMODE =  F24CE SC%ENT        - F5A73 NZ%HND(00958) Type=1.1 Nibs=4 Dist=035A5
CLOSEA =  120E4 TI%R6S        -
CLOSEF =  12087 TI%R6S        -
CLRFRC =  0C6F4 TI%R6S        -
CLRPRM =  04827 TI%R6S        -
CLRTSR =  0FD00 NZ%SYM        -
CMD1ST =  01654 TI%R6S        -
CMDFND =  01693 TI%R6S        -
CMDINI =  016D1 TI%R6S        -
CMDPR" =  01627 TI%R6S        -
CMDPTR =  2F6D4 TI%R6S        -
CMDS20 =  01672 TI%R6S        -
CMOSTV =  0168F TI%R6S        -
CMOSTW =  2F438 TI%R6S        -
CMPT   =  125B2 TI%R6S        -
CNFFND =  109AC TI%R6S        - F3C91 NZ%BUT(0009A) Type=0.1 Nibs=5
CNFLCT =  0BD15 TI%R6S        -
CNTADR =  2F67E TI%R6S        -
CNTRLd =  F7EA2 NZ%DEC        - F2A69 SC%ENT(00B35) Type=1.2 Nibs=5 Dist=05439
CNTRLp =  F7B9A NZ%PAR        - F2A6E SC%ENT(00B3A) Type=1.2 Nibs=5 Dist=0512C
CNVUCR =  152A7 TI%R6S        -
CNVWUC =  03FB8 TI%R6S        - F7BCE NZ%PAR(006D1) Type=0.1 Nibs=5
COLDST =  00000 TI%R6S        -
COLLAP =  091FB TI%R6S        -
COMCK  =  036CD TI%R6S        -
COMCK+ =  032AE TI%R6S        -
CONCOM =  0467E TI%R6S        -
CONF   =  10212 TI%R6S        -
CONFST =  2F9E6 TI%R6S        -
```

```
CONTRL =  F2A73 SC%ENT        - F01CA NZ%TBL(001C2) Type=1.2 Nibs=5 Dist=028A9
CONVUC =  F0E6D NZ%GPR        - F2E3C NZ%UTL(001A6) Type=1.1 Nibs=4 Dist=01FCF
CONWUC =  F7BCC NZ%PAR        -
COPYu  =  08269 TI%R6S        -
CORUPT =  09083 TI%R6S        -
COS12  =  0D721 TI%R6S        -
COS15  =  0D725 TI%R6S        -
COUNTC =  1C346 TI%R6S        - F270D SC%ENT(007D9) Type=0.1 Nibs=5
CPL#10 =  07887 TI%R6S        -
CPLXER =  F2631 SC%ENT        - F2629 SC%ENT(006F5) Type=1.2 Nibs=3 Dist=00008
CR     =  2C000 TI%R6S        -
CRDFIL =  1D21D TI%R6S        -
CREATE =  115A7 TI%R6S        -
CRETF+ =  084C4 TI%R6S        -
CRFSB- =  11664 TI%R6S        -
CRLFND =  0229E TI%R6S        - F664E NZ%CAT(007BD) Type=0.1 Nibs=5
CRLFOF =  02296 TI%R6S        -
CRLFSD =  022A2 TI%R6S        -
CRTF   =  116C1 TI%R6S        - F5975 NZ%HND(0085A) Type=0.1 Nibs=5
CSL9RO =  1BA0D TI%R6S        -
CSLC1  =  F0F42 NZ%GPR        -
CSLC10 =  F0F4A NZ%GPR        - F465F NZ%CAS(003CC) Type=1.1 Nibs=4 Dist=03715
                              + F5BDC NZ%HND(00AC1) Type=1.0 Nibs=4 Dist=04C92
CSLC11 =  F0F4D NZ%GPR        -
CSLC12 =  F0F50 NZ%GPR        - F1C1F NZ%BAS(00C85) Type=1.1 Nibs=4 Dist=00CCF
CSLC13 =  F0F53 NZ%GPR        -
CSLC14 =  F0F56 NZ%GPR        -
CSLC15 =  F0F59 NZ%GPR        -
CSLC2  =  F0F3F NZ%GPR        - F139B NZ%BAS(00401) Type=1.1 Nibs=3 Dist=0045C
                              + F4FC3 NZ%CAS(00D30) Type=1.1 Nibs=4 Dist=04084
                              + F5731 NZ%HND(00616) Type=1.1 Nibs=4 Dist=047F2
CSLC3  =  F0F3C NZ%GPR        - F12D3 NZ%BAS(00339) Type=1.1 Nibs=3 Dist=00397
                              + F4AA0 NZ%CAS(0080D) Type=1.0 Nibs=4 Dist=03B64
                              + F5462 NZ%HND(00347) Type=1.1 Nibs=4 Dist=04526
CSLC4  =  F0F39 NZ%GPR        - F1677 NZ%BAS(006DD) Type=1.0 Nibs=3 Dist=0073E
                              + F3E1C NZ%BUT(00225) Type=1.1 Nibs=4 Dist=02EE3
                              + F40FD NZ%BUT(00506) Type=1.0 Nibs=4 Dist=031C4
                              + F54B4 NZ%HND(00399) Type=1.0 Nibs=4 Dist=0457B
CSLC5  =  F0F36 NZ%GPR        - F2176 SC%ENT(00242) Type=1.1 Nibs=4 Dist=01240
                              + F217E SC%ENT(0024A) Type=1.1 Nibs=4 Dist=01248
                              + F274B SC%ENT(00817) Type=1.1 Nibs=4 Dist=01815
                              + F3092 NZ%BIF(001BB) Type=1.1 Nibs=4 Dist=0215C
                              + F66B6 NZ%CAT(00825) Type=1.0 Nibs=4 Dist=05780
                              + F7A9B NZ%PAR(0059E) Type=1.1 Nibs=4 Dist=06B65
CSLC6  =  F0F33 NZ%GPR        - F5286 NZ%HND(0016B) Type=1.1 Nibs=4 Dist=04353
CSLC7  =  F0F30 NZ%GPR        - F519F NZ%HND(00084) Type=1.1 Nibs=4 Dist=0426F
CSLC8  =  F0F2D NZ%GPR        - F4CBF NZ%CAS(00A2C) Type=1.1 Nibs=4 Dist=03D92
CSLC9  =  F0F47 NZ%GPR        - F3E80 NZ%BUT(00289) Type=1.1 Nibs=4 Dist=02F39
                              + F52E4 NZ%HND(001C9) Type=1.1 Nibs=4 Dist=0439D
CSLW3  =  0ED43 TI%R6S        -
CSLW4  =  0ED40 TI%R6S        -
CSLW5  =  0ED3D TI%R6S        -
CSPEED =  2F977 TI%R6S        -
CSRC1  =  F0F59 NZ%GPR        -
CSRC10 =  F0F33 NZ%GPR        - F5C69 NZ%HND(00B4E) Type=1.0 Nibs=4 Dist=04D36
                              + F66BC NZ%CAT(0082B) Type=1.0 Nibs=4 Dist=05789
CSRC11 =  F0F36 NZ%GPR        -
CSRC12 =  F0F39 NZ%GPR        - F6F05 NZ%FXQ(000EC) Type=1.1 Nibs=4 Dist=05FCC
CSRC13 =  F0F3C NZ%GPR        -
CSRC14 =  F0F3F NZ%GPR        -
```

```
CSRC15 =  F0F42 NZ%GPR        -
CSRC2  =  F0F56 NZ%GPR        - F4F63 NZ%CAS(00CD0) Type=1.1 Nibs=4 Dist=0400D
                              + F7130 NZ%FXQ(00317) Type=1.1 Nibs=4 Dist=061DA
CSRC3  =  F0F53 NZ%GPR        - F3EBB NZ%BUT(002C4) Type=1.1 Nibs=4 Dist=02F68
                              + F4A91 NZ%CAS(007FE) Type=1.0 Nibs=4 Dist=03B3E
                              + F547E NZ%HND(00363) Type=1.1 Nibs=4 Dist=0452B
CSRC4  =  F0F50 NZ%GPR        - F167E NZ%BAS(006E4) Type=1.0 Nibs=3 Dist=0072E
                              + F40F4 NZ%BUT(004FD) Type=1.0 Nibs=4 Dist=031A4
                              + F5257 NZ%HND(0013C) Type=1.1 Nibs=4 Dist=04307
                              + F52A5 NZ%HND(0018A) Type=1.1 Nibs=4 Dist=04355
CSRC5  =  F0F4D NZ%GPR        - F2195 SC%ENT(00261) Type=1.1 Nibs=4 Dist=01248
                              + F219E SC%ENT(0026A) Type=1.1 Nibs=4 Dist=01251
                              + F2586 SC%ENT(00652) Type=1.1 Nibs=4 Dist=01639
                              + F273B SC%ENT(00807) Type=1.1 Nibs=4 Dist=017EE
                              + F309F NZ%BIF(001C8) Type=1.1 Nibs=4 Dist=02152
                              + F5486 NZ%HND(0036B) Type=1.1 Nibs=4 Dist=04539
                              + F551C NZ%HND(00401) Type=1.1 Nibs=4 Dist=045CF
                              + F5726 NZ%HND(0060B) Type=1.1 Nibs=4 Dist=047D9
                              + F66AC NZ%CAT(0081B) Type=1.0 Nibs=4 Dist=0575F
                              + F7AB2 NZ%PAR(005B5) Type=1.1 Nibs=4 Dist=06B65
CSRC6  =  F0F4A NZ%GPR        -
CSRC7  =  F0F47 NZ%GPR        -
CSRC8  =  F0F2D NZ%GPR        - F49B6 NZ%CAS(00723) Type=1.1 Nibs=4 Dist=03A89
                              + F4B69 NZ%CAS(008D6) Type=1.1 Nibs=4 Dist=03C3C
CSRC9  =  F0F30 NZ%GPR        - F4DB6 NZ%CAS(00B23) Type=1.1 Nibs=4 Dist=03E86
                              + F53AE NZ%HND(00293) Type=1.1 Nibs=4 Dist=0447E
CSRW3  =  0ED32 TI%R6S        -
CSRW4  =  0ED2F TI%R6S        -
CSRW5  =  0ED2C TI%R6S        -
CURBOT =  10059 TI%R6S        -
CURDVC =  0A60B TI%R6S        -
CURREN =  2F56C TI%R6S        -
CURRL  =  2F7E8 TI%R6S        -
CURRST =  2F55D TI%R6S        -
CURSFL =  151DF TI%R6S        - F6647 NZ%CAT(007B6) Type=0.1 Nibs=5
CURSFR =  151D7 TI%R6S        -
CURSOR =  2F47E TI%R6S        - F3939 NZ%DSP(00302) Type=0.0 Nibs=4
                              + F3ACB NZ%DSP(00494) Type=0.0 Nibs=5
                              + F3BE4 NZ%DSP(005AD) Type=0.0 Nibs=5
CURSRD =  100A4 TI%R6S        -
CURSRT =  096C1 TI%R6S        -
CURSRU =  1009A TI%R6S        -
CURTOP =  10063 TI%R6S        -
CVUCW  =  03FBC TI%R6S        -
ChainE =  F7FFE Define        - F0028 NZ%TBL(00020) Type=1.2 Nibs=5 Dist=07FD6
Checks =  F7FFC Define        -
CkLoop =  1B669 TI%R6S        -
CkLpNC =  1B66D TI%R6S        -
CkTape =  00005 NZ%SYM        -
Clear  =  00005 TI%R6S        - F3BB9 NZ%DSP(00582) Type=0.0 Nibs=1
Clear? =  F3BAA NZ%DSP        -
CloseR =  00008 NZ%SYM        - F12EF NZ%BAS(00355) Type=0.0 Nibs=1
Cslc10 =  F5BDA NZ%HND        - F6091 NZ%CAT(00200) Type=1.1 Nibs=3 Dist=004B7
                              + F61D1 NZ%CAT(00340) Type=1.1 Nibs=3 Dist=005F7
                              + F6349 NZ%CAT(004B8) Type=1.1 Nibs=3 Dist=0076F
CurOff =  00006 TI%R6S        -

DO+2RD =  13A32 TI%R6S        -
DO=AVS =  09B2C TI%R6S        -
DO=FIB =  13AC5 TI%R6S        - F5433 NZ%HND(00318) Type=0.1 Nibs=5
```

```
D0=FRO  =  F5C9C NZ%HND        - F660A NZ%CAT(00779) Type=1.1 Nibs=4 Dist=0096E
                               + F6629 NZ%CAT(00798) Type=1.1 Nibs=4 Dist=0098D
D0=PCA  =  09B37 TI%R6S        - F2563 SC%ENT(0062F) Type=0.1 Nibs=5
D0@CUR  =  F3BCA NZ%DSP        -
D0ASC+  =  0982C TI%R6S        -
D0ASCI  =  09833 TI%R6S        -
D12ROA  =  1BA3C TI%R6S        -
D1=AVE  =  F0F74 NZ%GPR        - F2C0C SC%ENT(00CD8) Type=1.1 Nibs=4 Dist=01C98
                               + F3F54 NZ%BUT(0035D) Type=1.1 Nibs=4 Dist=02FE0
                               + F40CD NZ%BUT(004D6) Type=1.1 Nibs=4 Dist=03159
                               + F4614 NZ%CAS(00381) Type=1.1 Nibs=4 Dist=036A0
                               + F5FFB NZ%CAT(0016A) Type=1.1 Nibs=4 Dist=05087
D1=AVS  =  F0F7D NZ%GPR        - F253C SC%ENT(00608) Type=1.1 Nibs=4 Dist=015BF
                               + F6239 NZ%CAT(003A8) Type=1.1 Nibs=4 Dist=052BC
                               + F6ED1 NZ%FXQ(000B8) Type=1.1 Nibs=4 Dist=05F54
D1=DSP  =  F3281 NZ%BIF        - F1A04 NZ%BAS(00A6A) Type=1.1 Nibs=4 Dist=0187D
D1=DST  =  F328A NZ%BIF        - F11BF NZ%BAS(00225) Type=1.1 Nibs=4 Dist=020CB
                               + F195E NZ%BAS(009C4) Type=1.1 Nibs=4 Dist=0192C
                               + F1994 NZ%BAS(009FA) Type=1.1 Nibs=4 Dist=018F6
D1=DSX  =  F3293 NZ%BIF        - F11B4 NZ%BAS(0021A) Type=1.1 Nibs=4 Dist=020DF
D1=S20  =  F5C93 NZ%HND        -
D1=SCR  =  F4A2C NZ%CAS        - F5DC5 NZ%HND(00CAA) Type=1.1 Nibs=4 Dist=01399
D1=SDO  =  F1690 NZ%BAS        - F354F NZ%BIF(00678) Type=1.1 Nibs=4 Dist=01EBF
D1=SRO  =  F1687 NZ%BAS        -
D1@AVE  =  F0F86 NZ%GPR        - F2652 SC%ENT(0071E) Type=1.0 Nibs=4 Dist=016CC
                               + F3533 NZ%BIF(0065C) Type=1.1 Nibs=4 Dist=025AD
                               + F66C2 NZ%CAT(00831) Type=1.0 Nibs=4 Dist=0573C
                               + F6EE2 NZ%FXQ(000C9) Type=1.1 Nibs=4 Dist=05F5C
                               + F6F21 NZ%FXQ(00108) Type=1.1 Nibs=4 Dist=05F9B
                               + F74B8 NZ%FXQ(0069F) Type=1.1 Nibs=4 Dist=06532
D1@AVS  =  F0F92 NZ%GPR        - F46C4 NZ%CAS(00431) Type=1.1 Nibs=4 Dist=03732
                               + F6190 NZ%CAT(002FF) Type=1.1 Nibs=4 Dist=051FE
D1C=R3  =  03047 TI%R6S        -
D1FSTK  =  1955D TI%R6S        - F28EB SC%ENT(009B7) Type=0.1 Nibs=5
D1MST+  =  13E21 TI%R6S        - F2251 SC%ENT(0031D) Type=0.1 Nibs=5
D1MSTK  =  1954E TI%R6S        -
D=AVME  =  1A476 TI%R6S        -
D=AVMS  =  1A460 TI%R6S        -
D=WORD  =  04C0E TI%R6S        -
DATLEN  =  0B584 TI%R6S        -
DATPTR  =  2F692 TI%R6S        -
DAY2JD  =  13407 TI%R6S        -
DAYYMD  =  13335 TI%R6S        -
DBLPI4  =  0DAFC TI%R6S        -
DBLSUB  =  0DADD TI%R6S        -
DCHX=C  =  1B2D0 TI%R6S        -
DCHXF   =  1B223 TI%R6S        -
DCHXW   =  0ECDC TI%R6S        -
DCONTR  =  2E3FE TI%R6S        -
DCPLIN  =  10108 TI%R6S        -
DCRMNT  =  1C177 TI%R6S        - F2782 SC%ENT(0084E) Type=0.1 Nibs=5
DD1CTL  =  2E3FF TI%R6S        -
DD1END  =  2E34C TI%R6S        -
DD1ST   =  2E300 TI%R6S        -
DD2CTL  =  2E2FF TI%R6S        -
DD2END  =  2E260 TI%R6S        -
DD2ST   =  2E200 TI%R6S        -
DD3CTL  =  2E1FF TI%R6S        -
DD3END  =  2E160 TI%R6S        -
DD3ST   =  2E104 TI%R6S        -
```

```
DDL     =  F6BBA NZ%IOR         - F0BAE NZ%GPR(003EC) Type=1.1 Nibs=4 Dist=0600C
                                + F166F NZ%BAS(006D5) Type=1.0 Nibs=4 Dist=0554B
                                + F4A39 NZ%CAS(007A6) Type=1.0 Nibs=4 Dist=02181
                                + F5C8F NZ%HND(00B74) Type=1.0 Nibs=4 Dist=00F2B
DDT     =  F6BC9 NZ%IOR         - F0B60 NZ%GPR(0039E) Type=1.1 Nibs=4 Dist=06069
                                + F12FF NZ%BAS(00365) Type=1.0 Nibs=4 Dist=058CA
                                + F4A54 NZ%CAS(007C1) Type=1.0 Nibs=4 Dist=02175
                                + F6388 NZ%CAT(004F7) Type=1.1 Nibs=4 Dist=00841
DEBNCE  =  00CF7 TI%R6S         -
DECHEX  =  1B2D2 TI%R6S         -
DECP    =  0328F TI%R6S         -
DEFADC  =  052FC TI%R6S         -
DEFADR  =  2F967 TI%R6S         - F2C56 SC%ENT(00D22) Type=0.0 Nibs=5
DELAYT  =  2F948 TI%R6S         -
DELAYp  =  02AC6 TI%R6S         -
DEST    =  0F7B0 TI%R6S         -
DEVID   =  F1B39 NZ%BAS         - F00CE NZ%TBL(000C6) Type=1.2 Nibs=5 Dist=01A6B
DEVPAR  =  F1C85 NZ%BAS         -
DEVPR$  =  F1CCB NZ%BAS         - F60CC NZ%CAT(0023B) Type=1.1 Nibs=4 Dist=04401
DEVSPp  =  F78B2 NZ%PAR         - F06D1 NZ%DIR(0001C) Type=1.2 Nibs=5 Dist=071E1
DEVTYP  =  F1C3C NZ%BAS         - F00D7 NZ%TBL(000CF) Type=1.2 Nibs=5 Dist=01B65
DISINT  =  2F470 TI%R6S         -
DISPDC  =  05450 TI%R6S         - F7C2D NZ%DEC(0005A) Type=0.1 Nibs=5
DISPIS  =  F1142 NZ%BAS         - F0170 NZ%TBL(00168) Type=1.2 Nibs=5 Dist=00FD2
DISPP   =  035A4 TI%R6S         - F7518 NZ%PAR(0001B) Type=0.1 Nibs=5
DISPt   =  00000 TI%R6S         -
DIVF    =  0C4B8 TI%R6S         -
DMNSN   =  0AE39 TI%R6S         -
DONNA   =  09656 TI%R6S         -
DPART2  =  17EA3 TI%R6S         -
DPART3  =  17EF8 TI%R6S         -
DPOS    =  2F94D TI%R6S         -
DPVCTR  =  0AC50 TI%R6S         -
DRANGE  =  1B076 TI%R6S         -
DROPDC  =  05470 TI%R6S         -
DSLEEP  =  0056D TI%R6S         -
DSP$00  =  185DB TI%R6S         -
DSPBFE  =  2F540 TI%R6S         -
DSPBFS  =  2F480 TI%R6S         - F37B4 NZ%DSP(0017D) Type=0.0 Nibs=2 Offset=     -2
                                + F37D3 NZ%DSP(0019C) Type=0.0 Nibs=5
                                + F3ABF NZ%DSP(00488) Type=0.0 Nibs=5
DSPBUF  =  09723 TI%R6S         -
DSPCAT  =  F6606 NZ%CAT         -
DSPCHA  =  01C3E TI%R6S         -
DSPCHC  =  01C3C TI%R6S         -
DSPCHX  =  2F674 TI%R6S         - F3295 NZ%BIF(003BE) Type=0.0 Nibs=5
DSPCL?  =  020B6 TI%R6S         - F391C NZ%DSP(002E5) Type=0.1 Nibs=5
DSPCNA  =  09721 TI%R6S         -
DSPCNB  =  0971F TI%R6S         -
DSPCNO  =  09716 TI%R6S         -
DSPDGT  =  2F6DD TI%R6S         -
DSPFMT  =  2F6DC TI%R6S         -
DSPLI+  =  1010F TI%R6S         -
DSPLIN  =  10127 TI%R6S         -
DSPMSK  =  2F540 TI%R6S         -
DSPRST  =  02443 TI%R6S         -
DSPSET  =  2F7B1 TI%R6S         - F30A8 NZ%BIF(001D1) Type=0.0 Nibs=5
                                + F328C NZ%BIF(003B5) Type=0.0 Nibs=5
                                + F364F NZ%DSP(00018) Type=0.0 Nibs=4
                                + F36E0 NZ%DSP(000A9) Type=0.0 Nibs=4
```

```
                                + F392D NZ%DSP(002F6) Type=0.0 Nibs=4
                                + F3997 NZ%DSP(00360) Type=0.0 Nibs=5
                                + F3C61 NZ%BUT(0006A) Type=0.0 Nibs=5
DSPSTA =  2F475 TI%R6S          - F390F NZ%DSP(002D8) Type=0.0 Nibs=5
                                + F3923 NZ%DSP(002EC) Type=0.0 Nibs=4 Offset=      3
                                + F39D2 NZ%DSP(0039B) Type=0.0 Nibs=5 Offset=      3
                                + F3BAC NZ%DSP(00575) Type=0.0 Nibs=5 Offset=      3
DSPUPD =  01ADA TI%R6S          -
DSTRDC =  05280 TI%R6S          -
DTOH   =  F0D67 NZ%GPR          - F28A1 SC%ENT(0096D) Type=1.1 Nibs=4 Dist=01B3A
                                + F70E5 NZ%FXQ(002CC) Type=1.1 Nibs=4 Dist=0637E
                                + F712A NZ%FXQ(00311) Type=1.1 Nibs=4 Dist=063C3
                                + F7198 NZ%FXQ(0037F) Type=1.1 Nibs=4 Dist=06431
DV15M  =  0C4AC TI%R6S          -
DV15S  =  0C4B2 TI%R6S          -
DV2-12 =  0C4A8 TI%R6S          -
DV2-15 =  0C4AC TI%R6S          -
DVCPn* =  F79B0 NZ%PAR          -
DVCPy* =  F79B7 NZ%PAR          -
DVCSPp =  F79BA NZ%PAR          -
DVLBp  =  F788D NZ%PAR          -
DVSPp  =  F78B5 NZ%PAR          -
DVZNIB =  2F6FC TI%R6S          -
DWIDTH =  2F94F TI%R6S          -
DXP100 =  0CF7F TI%R6S          -
DZP    =  00003 TI%R6S          -
DdlPwr =  F5C73 NZ%HND          - F4EA3 NZ%CAS(00C10) Type=1.1 Nibs=4 Dist=00DD0
                                + F501C NZ%CAS(00D89) Type=1.1 Nibs=4 Dist=00C57
DdtRd  =  F4A3D NZ%CAS          - F5A27 NZ%HND(0090C) Type=1.1 Nibs=4 Dist=00FEA
DevID  =  0003F NZ%SYM          - F0A23 NZ%GPR(00261) Type=0.0 Nibs=2
                                + F3D86 NZ%BUT(0018F) Type=0.0 Nibs=2
                                + F7256 NZ%FXQ(0043D) Type=0.0 Nibs=2
DevTyp =  0001F NZ%SYM          - F09BF NZ%GPR(001FD) Type=0.0 Nibs=2
                                + F3D5E NZ%BUT(00167) Type=0.0 Nibs=2
                                + F4033 NZ%BUT(0043C) Type=0.0 Nibs=3
                                + F4250 NZ%BUT(00659) Type=0.0 Nibs=2
                                + F70F5 NZ%FXQ(002DC) Type=0.0 Nibs=2
Device =  0000A NZ%SYM          - F0C0B NZ%GPR(00449) Type=0.0 Nibs=1 Offset=     -8
                                + F0C62 NZ%GPR(004A0) Type=0.0 Nibs=1
Digit  =  00001 NZ%PAR          -
DispOK =  0000B NZ%SYM          - F2FAD NZ%BIF(000D6) Type=0.0 Nibs=1
                                + F3669 NZ%DSP(00032) Type=0.0 Nibs=1
                                + F368A NZ%DSP(00053) Type=0.0 Nibs=1
                                + F369A NZ%DSP(00063) Type=0.0 Nibs=1
                                + F36DD NZ%DSP(000A6) Type=0.0 Nibs=1
                                + F3994 NZ%DSP(0035D) Type=0.0 Nibs=1
                                + F3C6E NZ%BUT(00077) Type=0.0 Nibs=1
DsAddr =  00000 NZ%SYM          -
DsDevI =  00002 NZ%SYM          -
DsDevT =  00001 NZ%SYM          -
DsLoop =  00005 NZ%SYM          - F09B9 NZ%GPR(001F7) Type=0.0 Nibs=1 Offset=     -1
                                + F1F6A SC%ENT(00036) Type=0.0 Nibs=1
                                + F357D NZ%BIF(006A6) Type=0.0 Nibs=1
DsNull =  00004 NZ%SYM          - F09A3 NZ%GPR(001E1) Type=0.0 Nibs=1 Offset=     -1
                                + F3DF5 NZ%BUT(001FE) Type=0.0 Nibs=1
DsVolL =  00003 NZ%SYM          -

EDIT8O =  0A5A5 TI%R6S          -
EDITWF =  0A533 TI%R6S          -
EFIELD =  00000 TI%R6S          -
```

```
ENABLE =  F29DF SC%ENT        - F01B8 NZ%TBL(001B0) Type=1.2 Nibs=5 Dist=02827
ENABLd =  F7D22 NZ%DEC        - F29D5 SC%ENT(00AA1) Type=1.2 Nibs=5 Dist=0534D
ENABLp =  F7B47 NZ%PAR        - F29DA SC%ENT(00AA6) Type=1.2 Nibs=5 Dist=0516D
END    =  F0877 NZ%GPR        - F39CC NZ%DSP(00395) Type=1.1 Nibs=4 Dist=03155
ENDALL =  0769A TI%R6S        -
ENDBIN =  0764B TI%R6S        -
ENDFN  =  F0855 NZ%GPR        - F1B8A NZ%BAS(00BF0) Type=1.1 Nibs=4 Dist=01335
                              + F1C63 NZ%BAS(00CC9) Type=1.1 Nibs=4 Dist=0140E
ENDIMG =  1C040 TI%R6S        - F26F8 SC%ENT(007C4) Type=0.1 Nibs=5
ENDST  =  F084B NZ%GPR        - F1642 NZ%BAS(006A8) Type=1.0 Nibs=4 Dist=00DF7
                              + F2D70 NZ%UTL(000DA) Type=1.0 Nibs=4 Dist=02525
ENDSUB =  195A8 TI%R6S        -
ENDTAP =  F456E NZ%CAS        - F13E8 NZ%BAS(0044E) Type=1.1 Nibs=4 Dist=03186
                              + F5C2A NZ%HND(00B0F) Type=1.0 Nibs=4 Dist=016BC
ENTER  =  F1F58 SC%ENT        - F0143 NZ%TBL(0013B) Type=1.2 Nibs=5 Dist=01E15
ENTERp =  F751D NZ%PAR        - F1F53 SC%ENT(0001F) Type=1.2 Nibs=5 Dist=055CA
ENTUSG =  F2561 SC%ENT        - F075D NZ%DIR(000A8) Type=1.2 Nibs=5 Dist=01E04
EOLCK  =  02A7E TI%R6S        - F79BF NZ%PAR(004C2) Type=0.1 Nibs=5
                              + F7BB4 NZ%PAR(006B7) Type=0.1 Nibs=5
EOLCKR =  02A7A TI%R6S        -
EOLDC  =  05402 TI%R6S        -
EOLLEN =  2F95A TI%R6S        - F110D NZ%BAS(00173) Type=0.0 Nibs=5
                              + F2DB1 NZ%UTL(0011B) Type=0.0 Nibs=5
EOLSCN =  08AA7 TI%R6S        -
EOLSTR =  2F95B TI%R6S        -
EOLXC* =  052EC TI%R6S        -
EOLXCK =  05405 TI%R6S        -
ERR#   =  2F7E4 TI%R6S        -
ERRADR =  2F688 TI%R6S        -
ERRL#  =  2F7EC TI%R6S        -
ERRLCH =  2F97C TI%R6S        -
ERRM$f =  09806 TI%R6S        -
ERROR  =  F34E5 NZ%BIF        - F1F46 SC%ENT(00012) Type=1.0 Nibs=4 Dist=0159F
                              + F5CB9 NZ%HND(00B9E) Type=1.0 Nibs=4 Dist=027D4
ERROR! =  F34D8 NZ%BIF        - F7591 NZ%PAR(00094) Type=1.0 Nibs=4 Dist=040B9
ERRORP =  F34CE NZ%BIF        - F76C0 NZ%PAR(001C3) Type=1.0 Nibs=4 Dist=041F2
ERRORR =  F34CB NZ%BIF        - F7B0D NZ%PAR(00610) Type=1.0 Nibs=4 Dist=04642
ERRORX =  F34C1 NZ%BIF        - F1607 NZ%BAS(0066D) Type=1.0 Nibs=4 Dist=01EBA
                              + F1F74 SC%ENT(00040) Type=1.0 Nibs=4 Dist=0154D
                              + F2D38 NZ%UTL(000A2) Type=1.0 Nibs=3 Dist=00789
                              + F2E28 NZ%UTL(00192) Type=1.0 Nibs=3 Dist=00699
                              + F2EA5 NZ%UTL(0020F) Type=1.0 Nibs=3 Dist=0061C
                              + F3EED NZ%BUT(002F6) Type=1.0 Nibs=4 Dist=00A2C
                              + F52FE NZ%HND(001E3) Type=1.0 Nibs=4 Dist=01E3D
                              + F6DBC NZ%LOW(00066) Type=1.0 Nibs=4 Dist=038FB
ERRRTN =  074ED TI%R6S        -
ERRSUB =  2F683 TI%R6S        -
ESCSEQ =  023C1 TI%R6S        -
ESCSTA =  2F47B TI%R6S        - F372F NZ%DSP(000F8) Type=0.0 Nibs=5
EX-115 =  0CF48 TI%R6S        -
EX12   =  0D5C6 TI%R6S        -
EX15M  =  0D5CA TI%R6S        -
EX15S  =  0D5CE TI%R6S        -
EXAB1  =  0D3E7 TI%R6S        -
EXAB2  =  0D40E TI%R6S        -
EXACT  =  128B0 TI%R6S        -
EXCAD+ =  08631 TI%R6S        -
EXCHRe =  02E81 TI%R6S        -
EXCPAR =  187E8 TI%R6S        -
EXDCLP =  0592E TI%R6S        -
```

```
EXF    =  0D5DF TI%R6S        -
EXP15  =  0CF5A TI%R6S        -
EXPEX+ =  F4101 NZ%BUT        - F74F3 NZ%FXQ(006DA) Type=1.0 Nibs=4 Dist=033F2
EXPEX- =  0F178 TI%R6S        -
EXPEXC =  0F186 TI%R6S        - F4109 NZ%BUT(00512) Type=0.1 Nibs=5
EXPP10 =  03FE3 TI%R6S        -
EXPPAR =  03FD9 TI%R6S        - F7A62 NZ%PAR(00565) Type=0.1 Nibs=5
EXPPLS =  03FDC TI%R6S        -
EXPR   =  0F23C TI%R6S        -
EXPRDC =  05922 TI%R6S        - F7E73 NZ%DEC(002A0) Type=0.1 Nibs=5
EXPSKP =  1A9AC TI%R6S        -
EndNum =  000E6 TI%R6S        -
Endtap =  F5C28 NZ%HND        - F600C NZ%CAT(0017B) Type=1.1 Nibs=3 Dist=003E4
EolOK  =  00009 NZ%PAR        - F7705 NZ%PAR(00208) Type=0.0 Nibs=1
                              + F77E0 NZ%PAR(002E3) Type=0.0 Nibs=1
                              + F780E NZ%PAR(00311) Type=0.0 Nibs=1
Error  =  F5CB7 NZ%HND        - F5F20 NZ%CAT(0008F) Type=1.0 Nibs=3 Dist=00269
                              + F60DE NZ%CAT(0024D) Type=1.0 Nibs=3 Dist=00427
Except =  0000C TI%R6S        - F2B35 SC%ENT(00C01) Type=0.0 Nibs=1
                              + F318B NZ%BIF(002B4) Type=0.0 Nibs=1
ExprOK =  00008 NZ%PAR        - F76D5 NZ%PAR(001D8) Type=0.0 Nibs=1
                              + F77E3 NZ%PAR(002E6) Type=0.0 Nibs=1
                              + F77EB NZ%PAR(002EE) Type=0.0 Nibs=1

F->SCR =  F4A12 NZ%CAS        - F12C4 NZ%BAS(0032A) Type=1.1 Nibs=4 Dist=0374E
F-R0-0 =  2F89B TI%R6S        -
F-R0-1 =  2F8A0 TI%R6S        - F6151 NZ%CAT(002C0) Type=0.0 Nibs=5
                              + F6161 NZ%CAT(002D0) Type=0.0 Nibs=5
F-R0-2 =  2F8A5 TI%R6S        -
F-R0-3 =  2F8AA TI%R6S        -
F-R1-0 =  2F8AB TI%R6S        -
F-R1-1 =  2F8B0 TI%R6S        -
F-R1-2 =  2F8B5 TI%R6S        -
F-R1-3 =  2F8BA TI%R6S        -
FASCFD =  110C3 TI%R6S        -
FCHAIN =  F0008 Define        -
FCHLBL =  0782C TI%R6S        -
FCSTRT =  0E757 TI%R6S        -
FGTBL  =  00C9B TI%R6S        -
FIBAD- =  11478 TI%R6S        -
FIBADR =  11457 TI%R6S        -
FIBOFF =  12132 TI%R6S        - F3098 NZ%BIF(001C1) Type=0.1 Nibs=5
FILCRD =  1C879 TI%R6S        -
FILDC* =  05759 TI%R6S        - F7C4F NZ%DEC(0007C) Type=0.1 Nibs=5
FILEF  =  09FB0 TI%R6S        -
FILEP  =  03E9C TI%R6S        -
FILEP! =  03F0F TI%R6S        -
FILEP+ =  03F07 TI%R6S        -
FILEP- =  03F00 TI%R6S        -
FILEP1 =  03EFC TI%R6S        -
FILFIL =  011CE TI%R6S        -
FILSK+ =  06F1D TI%R6S        -
FILSPp =  F7862 NZ%PAR        - F06E0 NZ%DIR(0002B) Type=1.2 Nibs=5 Dist=07182
FILSPx =  F3528 NZ%BIF        - F06E5 NZ%DIR(00030) Type=1.2 Nibs=5 Dist=02E43
FILSp  =  F7857 NZ%PAR        -
FILXQ$ =  09B95 TI%R6S        -
FILXQ^ =  09B76 TI%R6S        -
FIND   =  F1BDB NZ%BAS        - F00C5 NZ%TBL(000BD) Type=1.2 Nibs=5 Dist=01B16
FINDA  =  023E3 TI%R6S        - F25CD SC%ENT(00699) Type=0.1 Nibs=5
                              + F3754 NZ%DSP(0011D) Type=0.1 Nibs=5
```

```
                                    + F5FC7 NZ%CAT(00136) Type=0.1 Nibs=5
FINDDO =  023E0 TI%R6S              -
FINDF  =  09F77 TI%R6S              - F562D NZ%HND(00512) Type=0.1 Nibs=5
                                    + F5929 NZ%HND(0080E) Type=0.1 Nibs=5
FINDF+ =  F473B NZ%CAS              - F5C6F NZ%HND(00B54) Type=1.0 Nibs=4 Dist=01534
FINDFL =  F4734 NZ%CAS              - F550F NZ%HND(003F4) Type=1.1 Nibs=4 Dist=00DDB
                                    + F57A6 NZ%HND(0068B) Type=1.1 Nibs=4 Dist=01072
FINDFx =  F47C7 NZ%CAS              - F5163 NZ%HND(00048) Type=1.1 Nibs=4 Dist=0099C
                                    + F5DB4 NZ%HND(00C99) Type=1.1 Nibs=4 Dist=015ED
FINDL  =  0FFE4 TI%R6S              -
FINDLB =  07786 TI%R6S              -
FINITA =  0CD03 TI%R6S              -
FINITC =  0CD0F TI%R6S              -
FINLIN =  18A3A TI%R6S              -
FIRSTC =  2F47C TI%R6S              -
FIXDC  =  05493 TI%R6S              -
FIXP   =  02A6E TI%R6S              -
FIXSPC =  00000 Define              - F11CA NZ%BAS(00230) Type=0.0 Nibs=1
                                    + F11E6 NZ%BAS(0024C) Type=0.0 Nibs=1
                                    + F133B NZ%BAS(003A1) Type=0.0 Nibs=1
                                    + F1CAB NZ%BAS(00D11) Type=0.0 Nibs=1
                                    + F1D37 NZ%BAS(00D9D) Type=0.0 Nibs=1
                                    + F222B SC%ENT(002F7) Type=0.0 Nibs=1
                                    + F226E SC%ENT(0033A) Type=0.0 Nibs=1
                                    + F24AB SC%ENT(00577) Type=0.0 Nibs=1
                                    + F2978 SC%ENT(00A44) Type=0.0 Nibs=1
                                    + F2A64 SC%ENT(00B30) Type=0.0 Nibs=1
                                    + F2C8F SC%ENT(00D5B) Type=0.0 Nibs=1
                                    + F44F8 NZ%CAS(00265) Type=0.0 Nibs=1
                                    + F6143 NZ%CAT(002B2) Type=0.0 Nibs=1
FLADDR =  0126B TI%R6S              -
FLDEVX =  01154 TI%R6S              -
FLGREG =  2F6E9 TI%R6S              -
FLIP10 =  0DB9C TI%R6S              -
FLIP11 =  0DBAB TI%R6S              -
FLIP8  =  0DB8D TI%R6S              -
FLOAT  =  1B322 TI%R6S              -
FLOAT! =  F6D56 NZ%LOW              - F1C5D NZ%BAS(00CC3) Type=1.1 Nibs=4 Dist=050F9
                                    + F1D87 NZ%BAS(00DED) Type=1.1 Nibs=4 Dist=04FCF
                                    + F1E77 NZ%BAS(00EDD) Type=1.1 Nibs=4 Dist=04EDF
FLOAT+ =  F6D59 NZ%LOW              -
FLOAT- =  F6D62 NZ%LOW              -
FLTDH  =  1B223 TI%R6S              - F1F2F NZ%BAS(00F95) Type=0.1 Nibs=5
FLTYPp =  03E71 TI%R6S              -
FNDCH- =  F0C10 NZ%GPR              - F15A7 NZ%BAS(0060D) Type=1.1 Nibs=4 Dist=00997
                                    + F2AC0 SC%ENT(00B8C) Type=1.1 Nibs=4 Dist=01EB0
FNDCHK =  F0C1B NZ%GPR              - F1DE2 NZ%BAS(00E48) Type=1.0 Nibs=4 Dist=011C7
                                    + F28F5 SC%ENT(009C1) Type=1.0 Nibs=4 Dist=01CDA
FNDCLR =  1DAEF TI%R6S              -
FNDFCN =  1A0A1 TI%R6S              -
FNDMB+ =  F3C3C NZ%BUT              - F560A NZ%HND(004EF) Type=1.1 Nibs=4 Dist=019CE
FNDMB- =  F3C40 NZ%BUT              - F0C12 NZ%GPR(00450) Type=1.1 Nibs=4 Dist=0302E
                                    + F6DEA NZ%LOW(00094) Type=1.1 Nibs=4 Dist=031AA
FNDMBD =  F3C5F NZ%BUT              - F2A98 SC%ENT(00B64) Type=1.1 Nibs=4 Dist=011C7
FNDMBX =  F3C75 NZ%BUT              - F0C1D NZ%GPR(0045B) Type=1.1 Nibs=4 Dist=03058
                                    + F2EFF NZ%BIF(00028) Type=1.1 Nibs=4 Dist=00D76
                                    + F3060 NZ%BIF(00189) Type=1.1 Nibs=4 Dist=00C15
                                    + F3149 NZ%BIF(00272) Type=1.1 Nibs=4 Dist=00B2C
                                    + F367E NZ%DSP(00047) Type=1.1 Nibs=3 Dist=005F7
FNPWDS =  0D3C0 TI%R6S              -
```

```
FNRTN1 =  0F216 TIXR6S        - F1D91 NZXBAS(00DF7) Type=0.1 Nibs=5
FNRTN2 =  0F219 TIXR6S        -
FNRTN3 =  0F235 TIXR6S        -
FNRTN4 =  0F238 TIXR6S        - F1E7D NZXBAS(00EE3) Type=0.1 Nibs=5
FORMAT =  F4326 NZXCAS        - F14FC NZXBAS(00562) Type=1.1 Nibs=4 Dist=02E2A
FORSTK =  2F59E TIXR6S        - F21C4 SCXENT(00290) Type=0.0 Nibs=5
                              + F21E2 SCXENT(002AE) Type=0.0 Nibs=5
FORUPD =  0A6AE TIXR6S        -
FPOLL  =  1250A TIXR6S        -
FRAC15 =  0C70E TIXR6S        -
FRAME+ =  F07C2 NZXGPR        - F683C NZXIOR(0016A) Type=1.1 Nibs=4 Dist=0607A
                              + F6866 NZXIOR(00194) Type=1.1 Nibs=4 Dist=060A4
                              + F6883 NZXIOR(001B1) Type=1.1 Nibs=4 Dist=060C1
FRAME- =  F07D0 NZXGPR        - F2411 SCXENT(004DD) Type=1.1 Nibs=4 Dist=01C41
                              + F5BAE NZXHND(00A93) Type=1.1 Nibs=4 Dist=053DE
                              + F6715 NZXIOR(00043) Type=1.1 Nibs=4 Dist=05F45
                              + F6950 NZXIOR(0027E) Type=1.1 Nibs=4 Dist=06180
FRAMEE =  F6BD8 NZXFRA        - F2CD5 NZXUTL(0003F) Type=1.1 Nibs=4 Dist=03F03
                              + F77BE NZXPAR(002C1) Type=1.1 Nibs=4 Dist=00BE6
FRAMET =  F6C81 NZXFRA        -
FRASPd =  F7D5E NZXDEC        -
FRASPp =  F7769 NZXPAR        -
FRange =  0B46A TIXR6S        -
FSPECe =  02F02 TIXR6S        -
FSPECp =  03CC5 TIXR6S        -
FSPECx =  09F2D TIXR6S        -
FTBSCH =  11093 TIXR6S        -
FTYPDC =  06902 TIXR6S        -
FTYPF# =  11059 TIXR6S        - F5CB2 NZXHND(00B97) Type=0.1 Nibs=5
FUNCD0 =  2F8BB TIXR6S        - F0FEC NZXBAS(00052) Type=0.0 Nibs=5
                              + F1002 NZXBAS(00068) Type=0.0 Nibs=5
                              + F1059 NZXBAS(000BF) Type=0.0 Nibs=5
                              + F33AD NZXBIF(004D6) Type=0.0 Nibs=5
                              + F33D9 NZXBIF(00502) Type=0.0 Nibs=5
FUNCD1 =  2F8C0 TIXR6S        - F33C3 NZXBIF(004EC) Type=0.0 Nibs=5
                              + F33EC NZXBIF(00515) Type=0.0 Nibs=5
                              + F3401 NZXBIF(0052A) Type=0.0 Nibs=5
                              + F3418 NZXBIF(00541) Type=0.0 Nibs=5
FUNCR0 =  2F89B TIXR6S        - F188A NZXBAS(008F0) Type=0.0 Nibs=5
                              + F1910 NZXBAS(00976) Type=0.0 Nibs=2
                              + F5949 NZXHND(0082E) Type=0.0 Nibs=2
                              + F5987 NZXHND(0086C) Type=0.0 Nibs=2 Offset=      5
                              + F599C NZXHND(00881) Type=0.0 Nibs=5 Offset=      5
                              + F5C9E NZXHND(00B83) Type=0.0 Nibs=5
FUNCR1 =  2F8AB TIXR6S        - F3433 NZXBIF(0055C) Type=0.0 Nibs=5
                              + F344D NZXBIF(00576) Type=0.0 Nibs=5
                              + F5936 NZXHND(0081B) Type=0.0 Nibs=2
                              + F697B NZXIOR(002A9) Type=0.0 Nibs=5
FXQPIL =  F73E4 NZXFXQ        -
FXQPn+ =  F742E NZXFXQ        -
FXQPnn =  F742B NZXFXQ        -
Findf+ =  F5C6D NZXHND        - F5E9E NZXCAT(0000D) Type=1.1 Nibs=3 Dist=00231
Format =  00005 NZXSYM        - F43E7 NZXCAS(00154) Type=0.0 Nibs=1

GADDR  =  F0994 NZXGPR        -
GADRR+ =  F404F NZXBUT        - F734F NZXFXQ(00536) Type=1.1 Nibs=4 Dist=03300
GADRRM =  F4040 NZXBUT        - F1CA1 NZXBAS(00D07) Type=1.1 Nibs=4 Dist=0239F
GADRST =  F70F9 NZXFXQ        -
GDIRSB =  F6346 NZXCAT        -
GDIRST =  F48D8 NZXCAS        - F1203 NZXBAS(00269) Type=1.1 Nibs=4 Dist=036D5
```

```
GDISP$ =  1C3C7 TI%R6S        -
GET    =  F67E6 NZ%IOR        - F0D14 NZ%GPR(00552) Type=1.0 Nibs=4 Dist=05AD2
GETALR =  F0EA2 NZ%GPR        - F4901 NZ%CAS(0066E) Type=1.1 Nibs=4 Dist=03A5F
                              + F491C NZ%CAS(00689) Type=1.1 Nibs=4 Dist=03A7A
                              + F494D NZ%CAS(006BA) Type=1.1 Nibs=4 Dist=03AAB
GETAVM =  1864D TI%R6S        -
GETBYT =  F50F6 NZ%CAS        - F584B NZ%HND(00730) Type=1.1 Nibs=3 Dist=00755
                              + F6470 NZ%CAT(005DF) Type=1.1 Nibs=4 Dist=0137A
GETCH# =  11427 TI%R6S        -
GETCON =  0DAA3 TI%R6S        -
GETD   =  F685D NZ%IOR        - F4840 NZ%CAS(005AD) Type=1.0 Nibs=4 Dist=0201D
                              + F5AAB NZ%HND(00990) Type=1.1 Nibs=4 Dist=00DB2
GETDID =  F6E19 NZ%FXQ        - F10EE NZ%BAS(00154) Type=1.1 Nibs=4 Dist=05D2B
                              + F1179 NZ%BAS(001DF) Type=1.1 Nibs=4 Dist=05CA0
                              + F11D7 NZ%BAS(0023D) Type=1.1 Nibs=4 Dist=05C42
                              + F132F NZ%BAS(00395) Type=1.1 Nibs=4 Dist=05AEA
                              + F15C1 NZ%BAS(00627) Type=1.1 Nibs=4 Dist=05858
                              + F1F5A SC%ENT(00026) Type=1.1 Nibs=4 Dist=04EBF
                              + F2A14 SC%ENT(00AE0) Type=1.1 Nibs=4 Dist=04405
GETDIM =  0AD6B TI%R6S        -
GETDIR =  F48B5 NZ%CAS        - F13DE NZ%BAS(00444) Type=1.1 Nibs=4 Dist=034D7
GETDIX =  F6E37 NZ%FXQ        - F1CF0 NZ%BAS(00D56) Type=1.1 Nibs=4 Dist=05147
GETDR! =  F486C NZ%CAS        - F5F50 NZ%CAT(000BF) Type=1.1 Nibs=4 Dist=016E4
                              + F6114 NZ%CAT(00283) Type=1.1 Nibs=4 Dist=018A8
GETDR" =  F4873 NZ%CAS        - F1218 NZ%BAS(0027E) Type=1.1 Nibs=4 Dist=0365B
                              + F1355 NZ%BAS(003BB) Type=1.1 Nibs=4 Dist=0351E
GETDR# =  F4875 NZ%CAS        -
GETDR+ =  F488E NZ%CAS        - F12A2 NZ%BAS(00308) Type=1.1 Nibs=4 Dist=035EC
                              + F6357 NZ%CAT(004C6) Type=1.1 Nibs=4 Dist=01AC9
GETDVW =  F71C8 NZ%FXQ        -
GETDev =  F0BF0 NZ%GPR        - F28FB SC%ENT(009C7) Type=1.0 Nibs=4 Dist=01D0B
                              + F4684 NZ%CAS(003F1) Type=1.1 Nibs=4 Dist=03A94
                              + F46D6 NZ%CAS(00443) Type=1.1 Nibs=4 Dist=03AE6
                              + F5AC5 NZ%HND(009AA) Type=1.1 Nibs=4 Dist=04ED5
                              + F5BC0 NZ%HND(00AA5) Type=1.1 Nibs=4 Dist=04FD0
                              + F5C1C NZ%HND(00B01) Type=1.1 Nibs=4 Dist=0502C
GETEND =  F687A NZ%IOR        -
GETERR =  F6826 NZ%IOR        - F08EB NZ%GPR(00129) Type=1.0 Nibs=4 Dist=05F3B
                              + F2D67 NZ%UTL(000D1) Type=1.1 Nibs=4 Dist=03ABF
                              + F2F0F NZ%BIF(00038) Type=1.1 Nibs=4 Dist=03917
                              + F362A NZ%BIF(00753) Type=1.0 Nibs=4 Dist=031FC
                              + F6DFE NZ%LOW(000A8) Type=1.1 Nibs=3 Dist=005D8
GETHEX =  F3FD1 NZ%BUT        - F14A8 NZ%BAS(0050E) Type=1.1 Nibs=4 Dist=02B29
GETHS2 =  F680C NZ%IOR        - F0C26 NZ%GPR(00464) Type=1.1 Nibs=4 Dist=05BE6
GETHSS =  F31CF NZ%BIF        - F1E4D NZ%BAS(00EB3) Type=1.1 Nibs=4 Dist=01382
                              + F2BB7 SC%ENT(00C83) Type=1.1 Nibs=3 Dist=00618
GETID  =  F68A3 NZ%IOR        - F0A69 NZ%GPR(002A7) Type=1.1 Nibs=4 Dist=05E3A
GETID+ =  F688F NZ%IOR        - F1B4A NZ%BAS(00BB0) Type=1.1 Nibs=4 Dist=04D45
GETLOP =  F2996 SC%ENT        - F6DE4 NZ%LOW(0008E) Type=1.1 Nibs=4 Dist=0444E
GETLPs =  F1DAA NZ%BAS        -
GETMBX =  F3BF7 NZ%BUT        - F0879 NZ%GPR(000B7) Type=1.0 Nibs=4 Dist=0337E
                              + F108C NZ%BAS(000F2) Type=1.1 Nibs=4 Dist=02B6B
                              + F162D NZ%BAS(00693) Type=1.1 Nibs=4 Dist=025CA
                              + F34FA NZ%BIF(00623) Type=1.1 Nibs=3 Dist=006FD
                              + F37E1 NZ%DSP(001AA) Type=1.1 Nibs=3 Dist=00416
                              + F3809 NZ%DSP(001D2) Type=1.1 Nibs=3 Dist=003EE
                              + F389C NZ%DSP(00265) Type=1.1 Nibs=3 Dist=0035B
                              + F3951 NZ%DSP(0031A) Type=1.1 Nibs=3 Dist=002A6
                              + F3982 NZ%DSP(0034B) Type=1.1 Nibs=3 Dist=00275
                              + F39AB NZ%DSP(00374) Type=1.1 Nibs=3 Dist=0024C
```

```
                                    + F3A1A NZ%DSP(003E3) Type=1.1 Nibs=3 Dist=001DD
                                    + F5427 NZ%HND(0030C) Type=1.0 Nibs=4 Dist=01830
                                    + F61E3 NZ%CAT(00352) Type=1.1 Nibs=4 Dist=025EC
                                    + F620B NZ%CAT(0037A) Type=1.1 Nibs=4 Dist=02614
                                    + F6A20 NZ%IOR(0034E) Type=1.1 Nibs=4 Dist=02E29
GETMSK =  01BBA TI%R6S              - F3B1D NZ%DSP(004E6) Type=0.1 Nibs=5
GETNAM =  1A085 TI%R6S              -
GETNE  =  F67D0 NZ%IOR              -
GETPI+ =  F6EA9 NZ%FXQ              - F356A NZ%BIF(00693) Type=1.1 Nibs=4 Dist=0393F
GETPIL =  F6EA0 NZ%FXQ              - F1458 NZ%BAS(004BE) Type=1.1 Nibs=4 Dist=05A48
GETPR1 =  06BFB TI%R6S              -
GETPR0 =  06BEE TI%R6S              -
GETSA  =  0E551 TI%R6S              -
GETST  =  F681C NZ%IOR              - F08F1 NZ%GPR(0012F) Type=1.1 Nibs=4 Dist=05F2B
                                    + F0C41 NZ%GPR(0047F) Type=1.1 Nibs=4 Dist=05BDB
                                    + F50C6 NZ%CAS(00E33) Type=1.1 Nibs=4 Dist=01756
GETST* =  07716 TI%R6S              -
GETST+ =  F3F41 NZ%BUT              -
GETST- =  F6833 NZ%IOR              - F3172 NZ%BIF(0029B) Type=1.1 Nibs=4 Dist=036C1
GETSTC =  07726 TI%R6S              -
GETSTR =  F3F19 NZ%BUT              - F19CF NZ%BAS(00A35) Type=1.1 Nibs=4 Dist=0254A
                                    + F6E1B NZ%FXQ(00002) Type=1.1 Nibs=4 Dist=02F02
                                    + F6EA2 NZ%FXQ(00089) Type=1.1 Nibs=4 Dist=02F89
GETVAL =  0DAB2 TI%R6S              -
GETX   =  F6745 NZ%IOR              - F23B2 SC%ENT(0047E) Type=1.1 Nibs=4 Dist=04393
                                    + F5BA0 NZ%HND(00A85) Type=1.1 Nibs=4 Dist=00BA5
GETZER =  F13F3 NZ%BAS              - F483A NZ%CAS(005A7) Type=1.0 Nibs=4 Dist=03447
GFTYPE =  F2E2B NZ%UTL              -
GHEXB+ =  F4016 NZ%BUT              - F1DC0 NZ%BAS(00E26) Type=1.1 Nibs=4 Dist=02256
GHEXBT =  F4012 NZ%BUT              - F2EAF NZ%UTL(00219) Type=1.1 Nibs=4 Dist=01163
                                    + F72BB NZ%FXQ(004A2) Type=1.1 Nibs=4 Dist=032A9
                                    + F7329 NZ%FXQ(00510) Type=1.1 Nibs=4 Dist=03317
                                    + F737A NZ%FXQ(00561) Type=1.1 Nibs=4 Dist=03368
GLOOP# =  F2DEF NZ%UTL              - F16B1 NZ%BAS(00717) Type=1.1 Nibs=4 Dist=0173E
                                    + F291A SC%ENT(009E6) Type=1.1 Nibs=3 Dist=004D5
                                    + F296D SC%ENT(00A39) Type=1.1 Nibs=3 Dist=00482
GNXTCR =  03064 TI%R6S              -
GOSUB  =  079E9 TI%R6S              -
GOSUBp =  029F6 TI%R6S              -
GOTO   =  079FA TI%R6S              -
GOTODC =  0552E TI%R6S              -
GOTOp  =  029F6 TI%R6S              -
GSBSTK =  2F5A3 TI%R6S              -
GST!NO =  F2E7C NZ%UTL              -
GT2BYO =  F50F2 NZ%CAS              - F64E9 NZ%CAT(00658) Type=1.1 Nibs=4 Dist=013F7
GT2BYT =  F50F4 NZ%CAS              - F1404 NZ%BAS(0046A) Type=1.0 Nibs=4 Dist=03CF0
                                    + F5C63 NZ%HND(00B48) Type=1.0 Nibs=4 Dist=00B6F
                                    + F63EA NZ%CAT(00559) Type=1.1 Nibs=4 Dist=012F6
GTEXT  =  05079 TI%R6S              -
GTEXT+ =  05199 TI%R6S              - F7CA7 NZ%DEC(000D4) Type=0.1 Nibs=5
GTEXT1 =  051A5 TI%R6S              -
GTFLAG =  1365E TI%R6S              -
GTKYC+ =  08D9B TI%R6S              -
GTKYCD =  08D92 TI%R6S              -
GTPTRS =  14636 TI%R6S              -
GTPTRX =  14670 TI%R6S              -
GTXT++ =  05192 TI%R6S              - F7EA4 NZ%DEC(002D1) Type=0.1 Nibs=5
GTYPE  =  F0C94 NZ%GPR              - F1C4A NZ%BAS(00CB0) Type=1.1 Nibs=4 Dist=00FB6
                                    + F369F NZ%DSP(00068) Type=1.1 Nibs=4 Dist=02A08
                                    + F42F3 NZ%CAS(00060) Type=1.1 Nibs=4 Dist=0365F
```

```
GTYPR+ =  F4001 NZ%BUT        - F295F SC%ENT(00A2B) Type=1.1 Nibs=4 Dist=016A2
                              + F298F SC%ENT(00A5B) Type=1.1 Nibs=4 Dist=01672
GTYPRM =  F4003 NZ%BUT        - F2E05 NZ%UTL(0016F) Type=1.1 Nibs=4 Dist=011FE
                              + F6DDB NZ%LOW(00085) Type=1.1 Nibs=4 Dist=02DD8
GTYPST =  F7088 NZ%FXQ        -
GetEXP =  1C086 TI%R6S        -
Getmbx =  F5425 NZ%HND        - F4C93 NZ%CAS(00A00) Type=1.1 Nibs=3 Dist=00792
                              + F4E42 NZ%CAS(00BAF) Type=1.1 Nibs=3 Dist=005E3

HASH1  =  1B0A1 TI%R6S        -
HASH2  =  1B0A3 TI%R6S        -
HDFLT  =  1B31B TI%R6S        - F27D3 SC%ENT(0089F) Type=0.1 Nibs=5
HEXASC =  17148 TI%R6S        -
HEXDEC =  0ECAF TI%R6S        -
HMSSEC =  13274 TI%R6S        -
HNDLFL =  0CBC9 TI%R6S        -
HPSCRH =  2F97F TI%R6S        -
HTOD   =  F0DBC NZ%GPR        - F1805 NZ%BAS(0086B) Type=1.1 Nibs=4 Dist=00A49
                              + F1BF0 NZ%BAS(00C56) Type=1.1 Nibs=4 Dist=00E34
                              + F1C09 NZ%BAS(00C6F) Type=1.1 Nibs=4 Dist=00E4D
HTODX  =  F0DE4 NZ%GPR        - F6412 NZ%CAT(00581) Type=1.1 Nibs=4 Dist=0562E
                              + F6516 NZ%CAT(00685) Type=1.1 Nibs=4 Dist=05732
                              + F6D5B NZ%LOW(00005) Type=1.1 Nibs=4 Dist=05F77
HTRAP  =  0CB2F TI%R6S        -
HUGE   =  0B75D TI%R6S        -
HXDASC =  05FF4 TI%R6S        -
HXDCW  =  0ECB4 TI%R6S        -

I/OAL+ =  1197B TI%R6S        -
I/OALL =  1197D TI%R6S        - F1A28 NZ%BAS(00A8E) Type=0.1 Nibs=5
                              + F2C39 SC%ENT(00D05) Type=0.1 Nibs=5
                              + F2EE5 NZ%BIF(0000E) Type=0.1 Nibs=5
                              + F3EA5 NZ%BUT(002AE) Type=0.1 Nibs=5
I/OCOL =  11979 TI%R6S        -
I/OCON =  11920 TI%R6S        -
I/ODAL =  11A41 TI%R6S        - F1B31 NZ%BAS(00B97) Type=0.1 Nibs=5
I/OEX2 =  11A0F TI%R6S        -
I/OEXP =  11A11 TI%R6S        -
I/OFND =  118BA TI%R6S        - F4110 NZ%BUT(00519) Type=0.1 Nibs=5
I/OFSC =  F362E NZ%IOB        - F3E92 NZ%BUT(0029B) Type=1.1 Nibs=4 Dist=00864
I/ORES =  118FF TI%R6S        - F302D NZ%BIF(00156) Type=0.1 Nibs=5
I/odal =  F1B2F NZ%BAS        - F3E6F NZ%BUT(00278) Type=1.1 Nibs=4 Dist=02340
IDIV   =  0EC7B TI%R6S        - F173D NZ%BAS(007A3) Type=0.1 Nibs=5
IDIVA  =  0EC6E TI%R6S        -
IF12A  =  0C739 TI%R6S        -
ILCNTe =  02E70 TI%R6S        -
IMDO+2 =  1BA2D TI%R6S        -
IMDO-2 =  1BA21 TI%R6S        -
IMerr  =  1B989 TI%R6S        - F2633 SC%ENT(006FF) Type=0.1 Nibs=5
IMinit =  1B88F TI%R6S        -
IMoffs =  1BA58 TI%R6S        -
IMxq27 =  1BB9C TI%R6S        -
INADDR =  2F6D4 TI%R6S        -
INBS   =  2F6C6 TI%R6S        -
INF*0  =  0C607 TI%R6S        -
INFR15 =  0C73D TI%R6S        -
INITD2 =  F7C6C NZ%DEC        -
INITFL =  F6979 NZ%IOR        - F50B9 NZ%CAS(00E26) Type=1.0 Nibs=4 Dist=018C0
INITIL =  F43F6 NZ%CAS        -
INITPR =  F75B8 NZ%PAR        -
```

```
INITXQ =  F1456 NZ%BAS        - F0104 NZ%TBL(000FC) Type=1.2 Nibs=5 Dist=01352
INITd  =  F7C3A NZ%DEC        - F144C NZ%BAS(004B2) Type=1.2 Nibs=5 Dist=067EE
INITp  =  F7571 NZ%PAR        - F1451 NZ%BAS(004B7) Type=1.2 Nibs=5 Dist=06120
INPOFF =  18B49 TI%R6S        -
INTA   =  2F410 TI%R6S        -
INTB   =  2F420 TI%R6S        -
INTGR  =  0F99B TI%R6S        -
INTM   =  2F430 TI%R6S        -
INTR4  =  2F400 TI%R6S        -
INTR50 =  000DB TI%R6S        -
INTRPT =  0000F TI%R6S        -
INVNaN =  0C65F TI%R6S        -
INXNIB =  2F6F9 TI%R6S        -
IOBFEN =  2F576 TI%R6S        -
IOBFST =  2F571 TI%R6S        -
IOFNDO =  118C1 TI%R6S        -
IOFSCR =  1188E TI%R6S        - F3632 NZ%IOB(00004) Type=0.1 Nibs=5
IOp    =  F765B NZ%PAR        - F17CE NZ%BAS(00834) Type=1.2 Nibs=5 Dist=05E8D
IS-DSP =  2F78D TI%R6S        - F1144 NZ%BAS(001AA) Type=0.0 Nibs=5
                              + F11A8 NZ%BAS(0020E) Type=0.0 Nibs=5
                              + F2FBC NZ%BIF(000E5) Type=0.0 Nibs=5 Offset=      3
                              + F30B6 NZ%BIF(001DF) Type=0.0 Nibs=4
                              + F3283 NZ%BIF(003AC) Type=0.0 Nibs=5
                              + F3639 NZ%DSP(00002) Type=0.0 Nibs=5
                              + F36EE NZ%DSP(000B7) Type=0.0 Nibs=2
                              + F3706 NZ%DSP(000CF) Type=0.0 Nibs=5 Offset=      3
                              + F3EF3 NZ%BUT(002FC) Type=0.0 Nibs=5 Offset=      3
IS-INP =  2F79B TI%R6S        -
IS-PLT =  2F7A2 TI%R6S        -
IS-PRT =  2F794 TI%R6S        - F0FA9 NZ%BAS(0000F) Type=0.0 Nibs=5
                              + F1166 NZ%BAS(001CC) Type=0.0 Nibs=5
IS-TBL =  2F78D TI%R6S        -
ISRAM? =  10192 TI%R6S        -
IVAERR =  0E920 TI%R6S        -
IVARG  =  0D749 TI%R6S        -
IVEXPe =  02E35 TI%R6S        -
IVLNIB =  2F6FD TI%R6S        -
IVP    =  00004 TI%R6S        -
IVPARe =  02E3F TI%R6S        -
IVVARe =  02E66 TI%R6S        -
InpByt =  00006 NZ%SYM        - F4495 NZ%CAS(00202) Type=0.0 Nibs=1
InhEOL =  00004 TI%R6S        -
Insert =  00007 TI%R6S        - F394D NZ%DSP(00316) Type=0.0 Nibs=1
                              + F396B NZ%DSP(00334) Type=0.0 Nibs=1
                              + F39EB NZ%DSP(003B4) Type=0.0 Nibs=1
                              + F3A41 NZ%DSP(0040A) Type=0.0 Nibs=1
                              + F3AA8 NZ%DSP(00471) Type=0.0 Nibs=1
InvalE =  00000 NZ%PAR        -

KCOL0  =  2F46F TI%R6S        -
KCOL1  =  2F46E TI%R6S        -
KCOL2  =  2F46D TI%R6S        -
KCOL3  =  2F46C TI%R6S        -
KCOL4  =  2F46B TI%R6S        -
KCOL5  =  2F46A TI%R6S        -
KCOL6  =  2F469 TI%R6S        -
KCOL7  =  2F468 TI%R6S        -
KCOL8  =  2F467 TI%R6S        -
KCOL9  =  2F466 TI%R6S        -
KCOLA  =  2F465 TI%R6S        -
```

```
KCOLB  =  2F464 TI%R6S        -
KCOLC  =  2F463 TI%R6S        -
KCOLD  =  2F462 TI%R6S        -
KEY$   =  1ACA8 TI%R6S        -
KEYBUF =  2F444 TI%R6S        -
KEYCOD =  1FD22 TI%R6S        -
KEYDEL =  08D2C TI%R6S        -
KEYFND =  08CB8 TI%R6S        -
KEYMRG =  08B8F TI%R6S        -
KEYNAM =  1AC04 TI%R6S        -
KEYPTR =  2F443 TI%R6S        - F31A8 NZ%BIF(002D1) Type=0.0 Nibs=5
KEYRD  =  14E11 TI%R6S        -
KEYSAV =  2F462 TI%R6S        -
KEYSCN =  00D4D TI%R6S        -
KYDN?  =  00774 TI%R6S        -

LABELP =  03E9F TI%R6S        -
LABLDC =  05702 TI%R6S        -
LASTFN =  000B4 TI%R6S        -
LBLIN# =  2F871 TI%R6S        -
LBLINP =  02A04 TI%R6S        -
LBLNAM =  077E7 TI%R6S        -
LBLNIF =  02A0D TI%R6S        -
LCDINI =  00665 TI%R6S        -
LDCEXT =  04F5E TI%R6S        -
LDCM10 =  04F6F TI%R6S        -
LDCOMP =  04F69 TI%R6S        -
LDCSET =  05060 TI%R6S        -
LDCSPC =  2F6C1 TI%R6S        -
LDSST1 =  04F72 TI%R6S        -
LDSST2 =  04F9E TI%R6S        -
LEAVE  =  04C01 TI%R6S        -
LEEWAY =  000D4 TI%R6S        -
LEXBF+ =  10DDF TI%R6S        - F5C06 NZ%HND(00AEB) Type=0.1 Nibs=5
LEXPIL =  000FF Define        - F1522 NZ%BAS(00588) Type=0.0 Nibs=2
                              + F3522 NZ%BIF(0064B) Type=0.0 Nibs=2
                              + F75F4 NZ%PAR(000F7) Type=0.0 Nibs=2
                              + F764E NZ%PAR(00151) Type=0.0 Nibs=2
                              + F7661 NZ%PAR(00164) Type=0.0 Nibs=2
                              + F767E NZ%PAR(00181) Type=0.0 Nibs=2
                              + F7B4D NZ%PAR(00650) Type=0.0 Nibs=2
                              + F7B79 NZ%PAR(0067C) Type=0.0 Nibs=2
                              + F7C99 NZ%DEC(000C6) Type=0.0 Nibs=2
LEXPTR =  2F6CF TI%R6S        - F7B1D NZ%PAR(00620) Type=0.0 Nibs=5
LGT15  =  0D1AE TI%R6S        -
LIMITS =  0AC3E TI%R6S        -
LIN#AU =  05122 TI%R6S        -
LIN#D+ =  05112 TI%R6S        -
LIN#DC =  05115 TI%R6S        -
LINEP  =  02620 TI%R6S        -
LINEP* =  02634 TI%R6S        -
LINEP+ =  02626 TI%R6S        -
LINP   =  02A07 TI%R6S        -
LISTDC =  05839 TI%R6S        -
LISTEN =  F0CF1 NZ%GPR        -
LISTID =  F17D3 NZ%BAS        - F0131 NZ%TBL(00129) Type=1.2 Nibs=5 Dist=016A2
LN1+15 =  0CD44 TI%R6S        -
LN1+XF =  0CD51 TI%R6S        -
LN12   =  0CD7D TI%R6S        -
LN15   =  0CD81 TI%R6S        -
```

```
LN30   =  0CD9C TI%R6S       -
LNEP66 =  027EA TI%R6S       -
LNPEXT =  02617 TI%R6S       -
LNSKP- =  089FF TI%R6S       -
LOCADR =  0A611 TI%R6S       -
LOCAL  =  F1517 NZ%BAS       - F0194 NZ%TBL(0018C) Type=1.2 Nibs=5 Dist=01383
LOCALd =  F7C8E NZ%DEC       - F150D NZ%BAS(00573) Type=1.2 Nibs=5 Dist=06781
LOCALp =  F75E9 NZ%PAR       - F1512 NZ%BAS(00578) Type=1.2 Nibs=5 Dist=060D7
LOCFIL =  1721D TI%R6S       -
LOCKWD =  2F7B2 TI%R6S       -
LOOP#d =  F7D3F NZ%DEC       -
LOOP#p =  F773C NZ%PAR       -
LOOPST =  2F7AC TI%R6S       - F0BF7 NZ%GPR(00435) Type=0.0 Nibs=5
                             + F0C55 NZ%GPR(00493) Type=0.0 Nibs=5
                             + F1948 NZ%BAS(009AE) Type=0.0 Nibs=5
                             + F1987 NZ%BAS(009ED) Type=0.0 Nibs=5
                             + F1A56 NZ%BAS(00ABC) Type=0.0 Nibs=5
                             + F2F44 NZ%BIF(0006D) Type=0.0 Nibs=2
                             + F3038 NZ%BIF(00161) Type=0.0 Nibs=4
                             + F30E9 NZ%BIF(00212) Type=0.0 Nibs=5
                             + F3C42 NZ%BUT(0004B) Type=0.0 Nibs=5
LSLEEP =  006CD TI%R6S       -
LSTCHR =  F3F92 NZ%BUT       -
LSTENT =  F4AC9 NZ%CAS       - F66CE NZ%CAT(0083D) Type=1.0 Nibs=4 Dist=01C05
LSTLEN =  06C27 TI%R6S       -
LXFND  =  0979D TI%R6S       -
LXTXTT =  1EE9F TI%R6S       -
Loop   =  0009F NZ%SYM       - F08A4 NZ%GPR(000E2) Type=0.0 Nibs=2
                             + F09AE NZ%GPR(001EC) Type=0.0 Nibs=2
                             + F724F NZ%FXQ(00436) Type=0.0 Nibs=2
LoopOK =  00008 NZ%SYM       - F1097 NZ%BAS(000FD) Type=0.0 Nibs=1
                             + F196A NZ%BAS(009D0) Type=0.0 Nibs=1
                             + F19A0 NZ%BAS(00A06) Type=0.0 Nibs=1
                             + F310A NZ%BIF(00233) Type=0.0 Nibs=1
                             + F365D NZ%DSP(00026) Type=0.0 Nibs=1
                             + F3991 NZ%DSP(0035A) Type=0.0 Nibs=1
                             + F3B79 NZ%DSP(00542) Type=0.0 Nibs=1
                             + F4A60 NZ%CAS(007CD) Type=0.0 Nibs=1
                             + F6A05 NZ%IOR(00333) Type=0.0 Nibs=1

MAIN05 =  00338 TI%R6S       -
MAIN30 =  0037E TI%R6S       -
MAINEN =  2F571 TI%R6S       -
MAINLP =  002FD TI%R6S       -
MAINST =  2F558 TI%R6S       -
MAKE1  =  0DACE TI%R6S       -
MAKEBF =  01751 TI%R6S       -
MAXCMD =  2F976 TI%R6S       -
MBOX^  =  2F7A9 TI%R6S       - F3138 NZ%BIF(00261) Type=0.0 Nibs=5
                             + F31BC NZ%BIF(002E5) Type=0.0 Nibs=5
                             + F3BF9 NZ%BUT(00002) Type=0.0 Nibs=5
                             + F3CD7 NZ%BUT(000E0) Type=0.0 Nibs=5
MEMBER =  1B098 TI%R6S       - F25BF SC%ENT(0068B) Type=0.1 Nibs=5
MEMCKL =  012A5 TI%R6S       -
MEMER* =  0945B TI%R6S       -
MEMERR =  0944D TI%R6S       - F28E4 SC%ENT(009B0) Type=0.1 Nibs=5
MEMERX =  0944F TI%R6S       -
MESSG  =  0CC17 TI%R6S       -
MFER42 =  0962C TI%R6S       -
MFERR  =  09393 TI%R6S       -
```

```
MFERR* =  093F1 TI%R6S        -
MFERRS =  0939E TI%R6S        -
MFERsp =  0940D TI%R6S        -
MFLG=0 =  13DA1 TI%R6S        - F22D8 SC%ENT(003A4) Type=0.1 Nibs=5
MFWRN  =  093BC TI%R6S        -
MFWRNQ =  093C5 TI%R6S        -
MFWRQ8 =  093C3 TI%R6S        -
MGOSUB =  1AF01 TI%R6S        -
MLFFLG =  2F870 TI%R6S        - F1CB3 NZ%BAS(00D19) Type=0.0 Nibs=5
                              + F1FB2 SC%ENT(0007E) Type=0.0 Nibs=5
                              + F229A SC%ENT(00366) Type=0.0 Nibs=5
MOVE*M =  01308 TI%R6S        -
MOVED0 =  1B0F4 TI%R6S        -
MOVED1 =  1B101 TI%R6S        -
MOVED2 =  1B104 TI%R6S        -
MOVED3 =  1B109 TI%R6S        -
MOVEDA =  1B0FA TI%R6S        -
MOVEDD =  1B106 TI%R6S        -
MOVEDM =  1B0EE TI%R6S        -
MOVEFL =  F4606 NZ%CAS        - F13C5 NZ%BAS(0042B) Type=1.1 Nibs=4 Dist=03241
                              + F55F3 NZ%HND(004D8) Type=1.1 Nibs=4 Dist=00FED
MOVEU0 =  1B162 TI%R6S        -
MOVEU1 =  1B16F TI%R6S        -
MOVEU2 =  1B172 TI%R6S        -
MOVEU3 =  1B177 TI%R6S        -
MOVEU4 =  1B174 TI%R6S        -
MOVEUA =  1B168 TI%R6S        -
MOVEUM =  1B15C TI%R6S        -
MP1-12 =  0C436 TI%R6S        -
MP15S  =  0C440 TI%R6S        -
MP2-12 =  0C432 TI%R6S        -
MP2-15 =  0C43A TI%R6S        -
MPOP1N =  0BD8D TI%R6S        -
MPOP2N =  0BD54 TI%R6S        -
MPY    =  0ECBB TI%R6S        -
MSN12  =  0D553 TI%R6S        -
MSN15  =  0D557 TI%R6S        -
MSPARe =  02E5C TI%R6S        -
MTADDR =  08195 TI%R6S        -
MTADR+ =  081A1 TI%R6S        -
MTHSTK =  2F599 TI%R6S        -
MTYL   =  F0D18 NZ%GPR        - F165D NZ%BAS(006C3) Type=1.0 Nibs=4 Dist=00945
                              + F36D4 NZ%DSP(0009D) Type=1.1 Nibs=4 Dist=029BC
                              + F4A7F NZ%CAS(007EC) Type=1.0 Nibs=4 Dist=03D67
                              + F5C87 NZ%HND(00B6C) Type=1.0 Nibs=4 Dist=04F6F
MTYLC  =  F0D26 NZ%GPR        -
MTYLL  =  F0D1F NZ%GPR        -
MULTF  =  0C446 TI%R6S        -
MVMEM+ =  0133C TI%R6S        -
MaxRec =  00007 NZ%SYM        - F435C NZ%CAS(000C9) Type=0.0 Nibs=1

NAMEp  =  F7A2D NZ%PAR        -
NAMEpb =  F7A29 NZ%PAR        -
NEEDSC =  2F94A TI%R6S        - F5FEB NZ%CAT(0015A) Type=0.0 Nibs=5
NEWFI+ =  F4ADE NZ%CAS        - F55BD NZ%HND(004A2) Type=1.1 Nibs=4 Dist=00ADF
                              + F5778 NZ%HND(0065D) Type=1.1 Nibs=4 Dist=00C9A
NEWFIL =  F4AFA NZ%CAS        - F52BA NZ%HND(0019F) Type=1.1 Nibs=3 Dist=007C0
NORAMe =  F3EE9 NZ%BUT        - F6254 NZ%CAT(003C3) Type=1.0 Nibs=4 Dist=0236B
NORDIM =  0AE2D TI%R6S        -
NOSCRL =  14C8A TI%R6S        -
```

```
NRMCON =  161AF TIXR6S        - F2145 SCXENT(00211) Type=0.1 Nibs=5
NTOKEN =  0493B TIXR6S        - F7B42 NZXPAR(00645) Type=0.1 Nibs=5
NTOKNL =  048E6 TIXR6S        -
NULLP  =  07999 TIXR6S        -
NUMC++ =  03690 TIXR6S        -
NUMC+0 =  03696 TIXR6S        -
NUMCK  =  F7AE4 NZXPAR        -
NUMCK+ =  F7AE0 NZXPAR        -
NUMSCN =  04D18 TIXR6S        - F212F SCXENT(001FB) Type=0.1 Nibs=5
NXTADR =  147E8 TIXR6S        -
NXTCHR =  F3F62 NZXBUT        - F1669 NZXBAS(006CF) Type=1.0 Nibs=4 Dist=028F9
                              + F749B NZXFXQ(00682) Type=1.0 Nibs=4 Dist=03539
NXTDST =  F2231 SCXENT        -
NXTELM =  148AC TIXR6S        -
NXTEN+ =  F4AB1 NZXCAS        -
NXTENT =  F4AB3 NZXCAS        - F124C NZXBAS(002B2) Type=1.1 Nibs=4 Dist=03867
                              + F1280 NZXBAS(002E6) Type=1.1 Nibs=4 Dist=03833
                              + F1428 NZXBAS(0048E) Type=1.1 Nibs=4 Dist=0368B
                              + F66C8 NZXCAT(00837) Type=1.0 Nibs=4 Dist=01C15
NXTEXP =  1C2F7 TIXR6S        - F26A3 SCXENT(0076F) Type=0.1 Nibs=5
NXTIRQ =  2F70D TIXR6S        -
NXTLIN =  10031 TIXR6S        -
NXTP   =  03455 TIXR6S        -
NXTSTM =  08A48 TIXR6S        - F1782 NZXBAS(007E8) Type=0.1 Nibs=5
NXTVA- =  13E58 TIXR6S        - F224A SCXENT(00316) Type=0.1 Nibs=5
NoCont =  0000E TIXR6S        -
Null   =  0007F NZXSYM        - F089B NZXGPR(000D9) Type=0.0 Nibs=2
                              + F0998 NZXGPR(001D6) Type=0.0 Nibs=2
                              + F723A NZXFXQ(00421) Type=0.0 Nibs=2
NumExp =  00003 NZXPAR        -
NwOFFS =  1C02D TIXR6S        -

OAGNXT =  03060 TIXR6S        -
OBCOLL =  01435 TIXR6S        -
OBEDIT =  17687 TIXR6S        -
OFFFLG =  2F442 TIXR6S        -
OFFIO  =  F192B NZXBAS        - F011F NZXTBL(00117) Type=1.2 Nibs=5 Dist=0180C
OFFIOd =  F7CD9 NZXDEC        - F17C9 NZXBAS(0082F) Type=1.2 Nibs=5 Dist=06510
                              + F1921 NZXBAS(00987) Type=1.2 Nibs=5 Dist=063B8
OFFIOp =  F7648 NZXPAR        - F1926 NZXBAS(0098C) Type=1.2 Nibs=5 Dist=05D22
OKP    =  00000 TIXR6S        -
ONDC20 =  05501 TIXR6S        - F7EC1 NZXDEC(002EE) Type=0.1 Nibs=5
ONINTR =  2F68D TIXR6S        - F1939 NZXBAS(0099F) Type=0.0 Nibs=5
                              + F29C6 SCXENT(00A92) Type=0.0 Nibs=5
                              + F2B26 SCXENT(00BF2) Type=0.0 Nibs=5
                              + F2B84 SCXENT(00C50) Type=0.0 Nibs=5
ONINTd =  F7EBB NZXDEC        - F29B5 SCXENT(00A81) Type=1.2 Nibs=5 Dist=05506
ONINTp =  F7678 NZXPAR        - F29BA SCXENT(00A86) Type=1.2 Nibs=5 Dist=04CBE
ONINTx =  F29BF SCXENT        - F014C NZXTBL(00144) Type=1.2 Nibs=5 Dist=02873
ONP40  =  02B7B TIXR6S        - F7697 NZXPAR(0019A) Type=0.1 Nibs=5
ONTIMR =  08008 TIXR6S        - F2B93 SCXENT(00C5F) Type=0.1 Nibs=5
OPENF  =  11B06 TIXR6S        -
ORGSB  =  0D65B TIXR6S        -
ORSB   =  0D63C TIXR6S        -
ORXM   =  0D633 TIXR6S        -
OUT1T+ =  02CDF TIXR6S        -
OUT1TK =  02CEB TIXR6S        - F7AC0 NZXPAR(005C3) Type=0.1 Nibs=5
OUT2TC =  02CFD TIXR6S        - F7AC7 NZXPAR(005CA) Type=0.1 Nibs=5
OUT2TK =  02CFF TIXR6S        -
OUT3TC =  F7ACC NZXPAR        - F7BDC NZXDEC(00009) Type=1.1 Nibs=3 Dist=00110
```

```
OUT3TK =  02D15 TI%R6S        - F7AD1 NZ%PAR(005D4) Type=0.1 Nibs=5
OUTBS  =  2F58F TI%R6S        -
OUTBY+ =  02CE5 TI%R6S        -
OUTBYT =  F7ABB NZ%PAR        - F7DCA NZ%DEC(001F7) Type=1.0 Nibs=3 Dist=0030F
OUTC15 =  05421 TI%R6S        -
OUTEL1 =  05300 TI%R6S        -
OUTELA =  05303 TI%R6S        - F7C60 NZ%DEC(0008D) Type=0.1 Nibs=5
OUTLI1 =  03709 TI%R6S        -
OUTLIT =  036F3 TI%R6S        -
OUTNBC =  F7AD6 NZ%PAR        - F7C4A NZ%DEC(00077) Type=1.1 Nibs=3 Dist=00174
                              + F7D1B NZ%DEC(00148) Type=1.1 Nibs=3 Dist=00245
                              + F7E8D NZ%DEC(002BA) Type=1.1 Nibs=3 Dist=003B7
                              + F7ED5 NZ%DEC(00302) Type=1.0 Nibs=3 Dist=003FF
OUTNBS =  05426 TI%R6S        - F7ADB NZ%PAR(005DE) Type=0.1 Nibs=5
OUTNIB =  02D28 TI%R6S        -
OUTPTt =  00002 NZ%SYM        - F10D0 NZ%BAS(00136) Type=0.0 Nibs=1
                              + F1126 NZ%BAS(0018C) Type=0.0 Nibs=1
                              + F5443 NZ%HND(00328) Type=0.0 Nibs=1
OUTPUT =  F10EC NZ%BAS        - F013A NZ%TBL(00132) Type=1.2 Nibs=5 Dist=00FB2
OUTPd  =  F7C06 NZ%DEC        - F10E2 NZ%BAS(00148) Type=1.2 Nibs=5 Dist=06B24
                              + F1F4E SC%ENT(0001A) Type=1.2 Nibs=5 Dist=05CB8
OUTPp  =  F750E NZ%PAR        - F10E7 NZ%BAS(0014D) Type=1.2 Nibs=5 Dist=06427
OUTRES =  0BC84 TI%R6S        -
OUTVAR =  0373E TI%R6S        -
OVFL   =  0CA73 TI%R6S        -
OVFNIB =  2F6FB TI%R6S        -
OVP    =  00002 TI%R6S        -
Offed  =  0000B NZ%SYM        - F1955 NZ%BAS(009BB) Type=0.0 Nibs=1
                              + F30F6 NZ%BIF(0021F) Type=0.0 Nibs=1
                              + F3C51 NZ%BUT(0005A) Type=0.0 Nibs=1
OptDev =  00008 NZ%PAR        - F79BC NZ%PAR(004BF) Type=0.0 Nibs=1
                              + F79C9 NZ%PAR(004CC) Type=0.0 Nibs=1
                              + F7B8E NZ%PAR(00691) Type=0.0 Nibs=1

P1-10  =  041C1 TI%R6S        -
PACK   =  F1346 NZ%BAS        - F0179 NZ%TBL(00171) Type=1.2 Nibs=5 Dist=011CD
PACKD  =  F11D5 NZ%BAS        - F0182 NZ%TBL(0017A) Type=1.2 Nibs=5 Dist=01053
PACKd  =  F7BDF NZ%DEC        - F11CB NZ%BAS(00231) Type=1.2 Nibs=5 Dist=06A14
                              + F133C NZ%BAS(003A2) Type=1.2 Nibs=5 Dist=068A3
PACKp  =  F79B0 NZ%PAR        - F11D0 NZ%BAS(00236) Type=1.2 Nibs=5 Dist=067E0
                              + F1341 NZ%BAS(003A7) Type=1.2 Nibs=5 Dist=0666F
PARERR =  02F08 TI%R6S        - F34E0 NZ%BIF(00609) Type=0.1 Nibs=5
PART3  =  18097 TI%R6S        -
PASS   =  F29FE SC%ENT        - F01AF NZ%TBL(001A7) Type=1.2 Nibs=5 Dist=0284F
PASSd  =  F7E78 NZ%DEC        - F29F4 SC%ENT(00AC0) Type=1.2 Nibs=5 Dist=05484
PASSp  =  F7B73 NZ%PAR        - F29F9 SC%ENT(00AC5) Type=1.2 Nibs=5 Dist=0517A
PCADDR =  2F679 TI%R6S        - F2A7C SC%ENT(00B48) Type=0.0 Nibs=5
PDEV   =  09E9E TI%R6S        -
PDIR   =  F11ED NZ%BAS        -
PEDIT  =  0FF5F TI%R6S        -
PEDITD =  0FF62 TI%R6S        -
PFINDL =  078DF TI%R6S        -
PFNDZL =  078E2 TI%R6S        -
PI/2   =  0DB77 TI%R6S        -
PI/2D  =  0DB7A TI%R6S        -
PI/4   =  0DAA1 TI%R6S        -
PILCNF =  F2F91 NZ%BIF        - F07A9 NZ%DIR(000F4) Type=1.2 Nibs=5 Dist=027E8
                              + F1152 NZ%BAS(001B8) Type=1.1 Nibs=4 Dist=01E3F
                              + F19B7 NZ%BAS(00A1D) Type=1.1 Nibs=4 Dist=015DA
PILCST =  F2ED7 NZ%BIF        - F0795 NZ%DIR(000E0) Type=1.2 Nibs=5 Dist=02742
```

```
PILDC  =  F7DA8 NZ%DEC        - F06D6 NZ%DIR(00021) Type=1.2 Nibs=5 Dist=076D2
PILMLP =  F30E7 NZ%BIF        - F07AE NZ%DIR(000F9) Type=1.2 Nibs=5 Dist=02939
PILMSG =  F040A Define        - F008C NZ%TBL(00084) Type=1.2 Nibs=4 Dist=0037E
PILPOF =  F3032 NZ%BIF        - F07A4 NZ%DIR(000EF) Type=1.2 Nibs=5 Dist=0288E
PILPOL =  F06B5 NZ%DIR        - F0090 NZ%TBL(00088) Type=1.2 Nibs=5 Dist=00625
PILSRQ =  F3119 NZ%BIF        - F07B3 NZ%DIR(000FE) Type=1.2 Nibs=5 Dist=02966
PILVER =  03142 Define        - F5149 NZ%HND(0002E) Type=0.0 Nibs=4
PILWKP =  F3019 NZ%BIF        - F079F NZ%DIR(000EA) Type=1.2 Nibs=5 Dist=0287A
PILWNK =  F2F68 NZ%BIF        - F079A NZ%DIR(000E5) Type=1.2 Nibs=5 Dist=027CE
PLOTt  =  00003 NZ%SYM        - F5448 NZ%HND(0032D) Type=0.0 Nibs=1
PNDALM =  2F761 TI%R6S        -
POLL   =  12337 TI%R6S        -
POLLD+ =  1232D TI%R6S        -
POP1N  =  F6D81 NZ%LOW        - F1789 NZ%BAS(007EF) Type=1.0 Nibs=4 Dist=055F8
                              + F60F4 NZ%CAT(00263) Type=1.1 Nibs=4 Dist=00C8D
POP1N+ =  0BD91 TI%R6S        -
POP1R  =  0E8FD TI%R6S        -
POP1S  =  0BD38 TI%R6S        - F3539 NZ%BIF(00662) Type=0.1 Nibs=5
                              + F60C5 NZ%CAT(00234) Type=0.1 Nibs=5
POP2N  =  0BC8C TI%R6S        - F1EF9 NZ%BAS(00F5F) Type=0.1 Nibs=5
POP2N+ =  0BD58 TI%R6S        -
POPBUF =  010EE TI%R6S        - F5F8C NZ%CAT(000FB) Type=0.1 Nibs=5
                              + F601E NZ%CAT(0018D) Type=0.1 Nibs=5
POPMTH =  1B3DB TI%R6S        - F21AB SC%ENT(00277) Type=0.1 Nibs=5
                              + F2284 SC%ENT(00350) Type=0.1 Nibs=5
POPSTK =  08F55 TI%R6S        -
POPSTR =  1B405 TI%R6S        -
POPUPD =  08F3E TI%R6S        - F74A7 NZ%FXQ(0068E) Type=0.1 Nibs=5
PPOS   =  2F956 TI%R6S        -
PRASCI =  F107F NZ%BAS        -
PREND  =  F10B7 NZ%BAS        - F107A NZ%BAS(000E0) Type=1.2 Nibs=5 Dist=0003D
PREP   =  0ADAF TI%R6S        -
PRESCN =  04A49 TI%R6S        -
PREXT  =  F0FD7 NZ%BAS        - F0FD2 NZ%BAS(00038) Type=1.2 Nibs=5 Dist=00005
PRGFMF =  0A146 TI%R6S        - F5B06 NZ%HND(009EB) Type=0.1 Nibs=5
PRGMEN =  2F567 TI%R6S        -
PRGMST =  2F562 TI%R6S        -
PRINT* =  17F37 TI%R6S        - F112F NZ%BAS(00195) Type=0.1 Nibs=5
PRINTt =  00001 TI%R6S        -
PRMCNT =  2F94B TI%R6S        -
PRMPTR =  2F5B7 TI%R6S        -
PRMSGA =  F0D4E NZ%GPR        - F4437 NZ%CAS(001A4) Type=1.1 Nibs=4 Dist=036E9
PRNEXe =  02E95 TI%R6S        -
PRNTOO =  F116B NZ%BAS        -
PRNTDC =  05450 TI%R6S        -
PRNTIS =  F1164 NZ%BAS        - F0167 NZ%TBL(0015F) Type=1.2 Nibs=5 Dist=00FFD
PRNTSd =  F7BD3 NZ%DEC        - F1138 NZ%BAS(0019E) Type=1.2 Nibs=5 Dist=06A9B
                              + F115A NZ%BAS(001C0) Type=1.2 Nibs=5 Dist=06A79
PRNTSp =  F74FD NZ%PAR        - F113D NZ%BAS(001A3) Type=1.2 Nibs=5 Dist=063C0
                              + F115F NZ%BAS(001C5) Type=1.2 Nibs=5 Dist=0639E
PROCDW =  F7215 NZ%FXQ        -
PROCLT =  F7263 NZ%FXQ        -
PROCST =  F6F50 NZ%FXQ        -
PRPSND =  06B17 TI%R6S        -
PRSCOO =  07B93 TI%R6S        -
PRSsc+ =  1BA84 TI%R6S        -
PRSscn =  1BA88 TI%R6S        -
PRT#DC =  06841 TI%R6S        -
PRTIS  =  F0FA1 NZ%BAS        - F0717 NZ%DIR(00062) Type=1.2 Nibs=5 Dist=0088A
PRTIS+ =  F0FAE NZ%BAS        - F5470 NZ%HND(00355) Type=1.1 Nibs=4 Dist=044C2
```

```
PRTISc =  F0F9A NZ%BAS        -
PSHGSB =  08F13 TI%R6S        -
PSHMCR =  08F0B TI%R6S        - F74E6 NZ%FXQ(006CD) Type=0.1 Nibs=5
PSHSTK =  08C7F TI%R6S        -
PSHSTL =  08C85 TI%R6S        -
PSHUPD =  08F0D TI%R6S        -
PT2BYT =  F510B NZ%CAS        - F13A0 NZ%BAS(00406) Type=1.1 Nibs=4 Dist=03D6B
                              + F5E78 NZ%HND(00D5D) Type=1.1 Nibs=4 Dist=00D6D
PUGFIB =  12198 TI%R6S        - F4E3C NZ%CAS(00BA9) Type=0.1 Nibs=5
PURFIB =  F5D39 NZ%HND        - F4C08 NZ%CAS(00975) Type=1.1 Nibs=4 Dist=01131
PURGDC =  05745 TI%R6S        -
PURGEF =  17359 TI%R6S        -
PUTALR =  F0ED2 NZ%GPR        - F4544 NZ%CAS(002B1) Type=1.1 Nibs=4 Dist=03672
PUTARL =  F0EBA NZ%GPR        -
PUTC   =  F6BB1 NZ%IOR        - F0CFC NZ%GPR(0053A) Type=1.0 Nibs=4 Dist=05EB5
                              + F1651 NZ%BAS(006B7) Type=1.0 Nibs=4 Dist=05560
                              + F24E2 SC%ENT(005AE) Type=1.0 Nibs=4 Dist=046CF
                              + F2D30 NZ%UTL(0009A) Type=1.1 Nibs=4 Dist=03E81
                              + F2DA7 NZ%UTL(00111) Type=1.1 Nibs=4 Dist=03E0A
                              + F31FF NZ%BIF(00328) Type=1.1 Nibs=4 Dist=039B2
                              + F4A5A NZ%CAS(007C7) Type=1.0 Nibs=4 Dist=02157
PUTC+  =  F6BAD NZ%IOR        - F0A3B NZ%GPR(00279) Type=1.1 Nibs=4 Dist=06172
                              + F307F NZ%BIF(001A8) Type=1.1 Nibs=4 Dist=03B2E
                              + F4E63 NZ%CAS(00BD0) Type=1.0 Nibs=4 Dist=01D4A
PUTC+N =  F6B7D NZ%IOR        -
PUTCN  =  F6B81 NZ%IOR        - F316C NZ%BIF(00295) Type=1.1 Nibs=4 Dist=03A15
PUTD   =  F6B43 NZ%IOR        - F0D0E NZ%GPR(0054C) Type=1.0 Nibs=4 Dist=05E35
                              + F2DD8 NZ%UTL(00142) Type=1.1 Nibs=4 Dist=03D6B
                              + F3BC6 NZ%DSP(0058F) Type=1.0 Nibs=4 Dist=02F7D
                              + F4A73 NZ%CAS(007E0) Type=1.0 Nibs=4 Dist=020D0
PUTDIR =  F5044 NZ%CAS        -
PUTDR" =  F5046 NZ%CAS        - F1329 NZ%BAS(0038F) Type=1.0 Nibs=4 Dist=03D1D
PUTDR# =  F5009 NZ%CAS        - F13AF NZ%BAS(00415) Type=1.1 Nibs=4 Dist=03C5A
                              + F5D0C NZ%HND(00BF1) Type=1.0 Nibs=4 Dist=00D03
PUTDX  =  F0EEA NZ%GPR        - F4A79 NZ%CAS(007E6) Type=1.0 Nibs=4 Dist=03B8F
PUTE   =  F6B55 NZ%IOR        - F08D9 NZ%GPR(00117) Type=1.1 Nibs=4 Dist=0627C
                              + F0CA7 NZ%GPR(004E5) Type=1.1 Nibs=4 Dist=05EAE
                              + F1779 NZ%BAS(007DF) Type=1.1 Nibs=4 Dist=053DC
                              + F24F9 SC%ENT(005C5) Type=1.0 Nibs=4 Dist=0465C
                              + F327D NZ%BIF(003A6) Type=1.0 Nibs=4 Dist=038D8
                              + F42A6 NZ%CAS(00013) Type=1.1 Nibs=4 Dist=028AF
                              + F436B NZ%CAS(000D8) Type=1.1 Nibs=4 Dist=027EA
                              + F44AA NZ%CAS(00217) Type=1.1 Nibs=4 Dist=026AB
                              + F5AA2 NZ%HND(00987) Type=1.1 Nibs=4 Dist=010B3
                              + F5B92 NZ%HND(00A77) Type=1.1 Nibs=4 Dist=00FC3
PUTEN  =  F6B86 NZ%IOR        -
PUTEX  =  F6B5D NZ%IOR        -
PUTEsc =  F3BC0 NZ%DSP        -
PUTGF  =  F0C86 NZ%GPR        - F2A4D SC%ENT(00B19) Type=1.1 Nibs=4 Dist=01DC7
PUTGF+ =  F0C82 NZ%GPR        -
PUTGF- =  F0C7E NZ%GPR        - F1D56 NZ%BAS(00DBC) Type=1.1 Nibs=4 Dist=010D8
PUTRES =  18115 TI%R6S        -
PUTX   =  F6A97 NZ%IOR        - F3816 NZ%DSP(001DF) Type=1.1 Nibs=4 Dist=03281
                              + F3961 NZ%DSP(0032A) Type=1.1 Nibs=4 Dist=03136
PWIDTH =  2F958 TI%R6S        -
PWROFF =  00526 TI%R6S        -
PWrite =  00006 NZ%SYM        - F5C74 NZ%HND(00B59) Type=0.0 Nibs=1
PgmRun =  0000D TI%R6S        -
Positn =  00003 NZ%SYM        -
Printr =  00009 NZ%SYM        - F3673 NZ%DSP(0003C) Type=0.0 Nibs=1
```

```
                                + F36AB NZ%DSP(00074) Type=0.0 Nibs=1
                                + F36C0 NZ%DSP(00089) Type=0.0 Nibs=1
                                + F3740 NZ%DSP(00109) Type=0.0 Nibs=1
                                + F379A NZ%DSP(00163) Type=0.0 Nibs=1
Putd    =  F0D0C NZ%GPR         - F131F NZ%BAS(00385) Type=1.1 Nibs=3 Dist=00613
Pute    =  F327B NZ%BIF         - F2CC3 NZ%UTL(0002D) Type=1.1 Nibs=3 Dist=005B8

QUOEXe  =  02E8B TI%R6S         -
QUOTCK  =  0623D TI%R6S         -

R1REV   =  00785 TI%R6S         -
R2REV   =  0AA83 TI%R6S         -
R3=D10  =  03526 TI%R6S         -
R3REV   =  153AB TI%R6S         -
R4REV   =  1DBA8 TI%R6S         -
R<RST2  =  014DB TI%R6S         - F61DC NZ%CAT(0034B) Type=0.1 Nibs=5
R<RSTK  =  014DD TI%R6S         -
RAMEND  =  2F5B2 TI%R6S         -
RAMROM  =  0A5F7 TI%R6S         -
RANGE   =  F0E93 NZ%GPR         -
RANGEA  =  F0E83 NZ%GPR         - F2E45 NZ%UTL(001AF) Type=1.1 Nibs=4 Dist=01FC2
                                + F7D73 NZ%DEC(001A0) Type=1.1 Nibs=4 Dist=06EF0
RANGEN  =  F0E8D NZ%GPR         - F20F9 SC%ENT(001C5) Type=1.1 Nibs=4 Dist=0126C
                                + F74CA NZ%FXQ(006B1) Type=1.0 Nibs=4 Dist=0663D
RAWBFR  =  2F580 TI%R6S         -
RCCD1   =  0D3F5 TI%R6S         -
RCCD2   =  0D41C TI%R6S         -
RCL*    =  0E983 TI%R6S         -
RCLW1   =  0E981 TI%R6S         -
RCLW2   =  0E9BE TI%R6S         -
RCLW3   =  0E9C4 TI%R6S         -
RCSCR   =  0E954 TI%R6S         -
RCVOFS  =  1C050 TI%R6S         - F26CC SC%ENT(00798) Type=0.1 Nibs=5
RDATTY  =  17CC6 TI%R6S         - F2269 SC%ENT(00335) Type=0.1 Nibs=5
RDBAS   =  173FF TI%R6S         -
RDBYTA  =  13A2F TI%R6S         -
RDCHD+  =  076EE TI%R6S         -
RDCHDR  =  076F0 TI%R6S         -
RDHDR1  =  076FD TI%R6S         -
RDINFO  =  F4254 NZ%BUT         - F5C5B NZ%HND(00B40) Type=1.0 Nibs=4 Dist=01A07
RDLNAS  =  13A1F TI%R6S         -
RDTEXT  =  17489 TI%R6S         -
READDC  =  F1D99 NZ%BAS         - F00F2 NZ%TBL(000EA) Type=1.2 Nibs=5 Dist=01CA7
READI3  =  F6736 NZ%IOR         - F44FB NZ%CAS(00268) Type=1.1 Nibs=4 Dist=0223B
READIN  =  F1D46 NZ%BAS         - F00E9 NZ%TBL(000E1) Type=1.2 Nibs=5 Dist=01C5D
READIT  =  F66DE NZ%IOR         -
READNB  =  17518 TI%R6S         -
READP5  =  0323B TI%R6S         - F7542 NZ%PAR(00045) Type=0.1 Nibs=5
READR#  =  F4594 NZ%CAS         - F52D8 NZ%HND(001BD) Type=1.1 Nibs=4 Dist=00D44
                                + F5361 NZ%HND(00246) Type=1.1 Nibs=4 Dist=00DCD
READRG  =  F689A NZ%IOR         - F0B7A NZ%GPR(003B8) Type=1.1 Nibs=4 Dist=05D20
                                + F1BC0 NZ%BAS(00C26) Type=1.1 Nibs=4 Dist=04CDA
READSU  =  F66D2 NZ%IOR         - F4A6D NZ%CAS(007DA) Type=1.0 Nibs=4 Dist=01C65
                                + F5A8A NZ%HND(0096F) Type=1.1 Nibs=4 Dist=00C48
RECADR  =  0F4B7 TI%R6S         -
RECALL  =  0F281 TI%R6S         -
REDCHR  =  F22F7 SC%ENT         -
REDUCE  =  15977 TI%R6S         -
RELJMP  =  05047 TI%R6S         -
REMOTE  =  F1570 NZ%BAS         - F019D NZ%TBL(00195) Type=1.2 Nibs=5 Dist=013D3
```

```
REMOTd =  F7CC7 NZ%DEC        - F1566 NZ%BAS(005CC) Type=1.2 Nibs=5 Dist=06761
REMOTp =  F761E NZ%PAR        - F156B NZ%BAS(005D1) Type=1.2 Nibs=5 Dist=060B3
RENSUB =  1A753 TI%R6S        -
REPROM =  18A1E TI%R6S        -
REQST  =  F2919 SC%ENT        - F018B NZ%TBL(00183) Type=1.2 Nibs=5 Dist=0278E
REQSTd =  F7D29 NZ%DEC        - F290F SC%ENT(009DB) Type=1.2 Nibs=5 Dist=0541A
REQSTp =  F7B5D NZ%PAR        - F2914 SC%ENT(009E0) Type=1.2 Nibs=5 Dist=05249
RESCAN =  04A4C TI%R6S        -
RESERV =  2F986 TI%R6S        -
RESET  =  F6DCA NZ%LOW        - F015E NZ%TBL(00156) Type=1.2 Nibs=5 Dist=06C6C
RESETd =  F7D0E NZ%DEC        - F6DC0 NZ%LOW(0006A) Type=1.2 Nibs=5 Dist=00F4E
RESETp =  F7628 NZ%PAR        - F6DC5 NZ%LOW(0006F) Type=1.2 Nibs=5 Dist=00863
RESPTR =  F7B1B NZ%PAR        -
RESREG =  2F7C2 TI%R6S        -
RESST+ =  F349C NZ%BIF        -
RESSTS =  F348E NZ%BIF        - F2B02 SC%ENT(00BCE) Type=1.1 Nibs=4 Dist=0098C
REST*  =  03035 TI%R6S        - F766E NZ%PAR(00171) Type=0.1 Nibs=5
REST10 =  F1985 NZ%BAS        - F2AD9 SC%ENT(00BA5) Type=1.0 Nibs=4 Dist=01154
REST1A =  F337E NZ%BIF        - F14D3 NZ%BAS(00539) Type=1.1 Nibs=4 Dist=01EAB
REST2C =  F3392 NZ%BIF        - F14C3 NZ%BAS(00529) Type=1.1 Nibs=4 Dist=01ECF
                              + F16FB NZ%BAS(00761) Type=1.1 Nibs=4 Dist=01C97
                              + F172A NZ%BAS(00790) Type=1.1 Nibs=4 Dist=01C68
                              + F2D0C NZ%UTL(00076) Type=1.1 Nibs=3 Dist=00686
                              + F2D7B NZ%UTL(000E5) Type=1.1 Nibs=3 Dist=00617
                              + F72EB NZ%FXQ(004D2) Type=1.1 Nibs=4 Dist=03F59
                              + F73A9 NZ%FXQ(00590) Type=1.1 Nibs=4 Dist=04017
RESTDO =  F32F9 NZ%BIF        - F117F NZ%BAS(001E5) Type=1.1 Nibs=4 Dist=0217A
                              + F2076 SC%ENT(00142) Type=1.1 Nibs=4 Dist=01283
                              + F2968 SC%ENT(00A34) Type=1.0 Nibs=4 Dist=00991
                              + F2D62 NZ%UTL(000CC) Type=1.1 Nibs=3 Dist=00597
RESTD1 =  F330C NZ%BIF        - F14B8 NZ%BAS(0051E) Type=1.1 Nibs=4 Dist=01E54
                              + F28B2 SC%ENT(0097E) Type=1.1 Nibs=4 Dist=00A5A
                              + F5F3C NZ%CAT(000AB) Type=1.1 Nibs=4 Dist=02C30
                              + F5FA9 NZ%CAT(00118) Type=1.1 Nibs=4 Dist=02C9D
                              + F72E2 NZ%FXQ(004C9) Type=1.1 Nibs=4 Dist=03FD6
                              + F739D NZ%FXQ(00584) Type=1.1 Nibs=4 Dist=04091
RESTIO =  F197F NZ%BAS        - F0128 NZ%TBL(00120) Type=1.2 Nibs=5 Dist=01857
RESTOR =  F3EF1 NZ%BUT        - F2FB6 NZ%BIF(000DF) Type=1.1 Nibs=4 Dist=00F3B
RESTRT =  F308D NZ%BIF        - F0916 NZ%GPR(00154) Type=1.1 Nibs=4 Dist=02777
                              + F0923 NZ%GPR(00161) Type=1.1 Nibs=4 Dist=0276A
RESTST =  F32B7 NZ%BIF        - F4117 NZ%BUT(00520) Type=1.0 Nibs=4 Dist=00E60
                              + F74C4 NZ%FXQ(006AB) Type=1.0 Nibs=4 Dist=0420D
RESTd  =  F7CFE NZ%DEC        - F1975 NZ%BAS(009DB) Type=1.2 Nibs=5 Dist=06389
RESTp  =  F7BAE NZ%PAR        - F197A NZ%BAS(009E0) Type=1.2 Nibs=5 Dist=06234
REV$   =  1B38E TI%R6S        - F61B4 NZ%CAT(00323) Type=0.1 Nibs=5
REVPOP =  0BD31 TI%R6S        - F1CC6 NZ%BAS(00D2C) Type=0.1 Nibs=5
                              + F3F43 NZ%BUT(0034C) Type=0.1 Nibs=5
REWIND =  11365 TI%R6S        -
RFAD++ =  0A6FB TI%R6S        -
RFAD+I =  0A702 TI%R6S        -
RFAD-- =  0A652 TI%R6S        -
RFAD-I =  0A659 TI%R6S        -
RFNBFR =  2F57B TI%R6S        -
RFUPD+ =  0A66E TI%R6S        -
RJUST  =  12AE2 TI%R6S        -
RND-12 =  1B01F TI%R6S        -
RND12+ =  0C9D5 TI%R6S        -
RNDAHX =  136CB TI%R6S        -
RNDNRM =  0CAB1 TI%R6S        -
RNSEED =  2F6FE TI%R6S        -
```

```
ROMCID =  00BFE TI%R6S        -
ROMFND =  1102F TI%R6S        -
ROMSTT =  F0000 NZ%RST        -
ROMTYP =  F4167 NZ%BUT        - F7220 NZ%FXQ(00407) Type=1.1 Nibs=4 Dist=030B9
ROWDVR =  2E350 TI%R6S        -
RPLLIN =  013F7 TI%R6S        -
RPLSBH =  1799B TI%R6S        -
RPTKY  =  152BA TI%R6S        - F5F93 NZ%CAT(00102) Type=0.1 Nibs=5
RSDOD1 =  F7AAA NZ%PAR        -
RST2<R =  014A6 TI%R6S        - F6204 NZ%CAT(00373) Type=0.1 Nibs=5
RSTK<R =  014A8 TI%R6S        -
RSTKBF =  2F820 TI%R6S        -
RSTKBp =  2F81F TI%R6S        -
RSTST  =  0F5C5 TI%R6S        -
RTNCC  =  F79AE NZ%PAR        -
RTNCCX =  F2F62 NZ%BIF        -
RTNSXM =  F078F NZ%DIR        - F06DB NZ%DIR(00026) Type=1.2 Nibs=5 Dist=000B4
                              + F0708 NZ%DIR(00053) Type=1.2 Nibs=5 Dist=00087
                              + F070D NZ%DIR(00058) Type=1.2 Nibs=5 Dist=00082
                              + F072B NZ%DIR(00076) Type=1.2 Nibs=5 Dist=00064
                              + F0730 NZ%DIR(0007B) Type=1.2 Nibs=5 Dist=0005F
                              + F0735 NZ%DIR(00080) Type=1.2 Nibs=5 Dist=0005A
                              + F073A NZ%DIR(00085) Type=1.2 Nibs=5 Dist=00055
                              + F0758 NZ%DIR(000A3) Type=1.2 Nibs=5 Dist=00037
RUNRT1 =  074E7 TI%R6S        -
RUNRTN =  074EA TI%R6S        -
Read   =  00002 NZ%SYM        - F4A3E NZ%CAS(007AB) Type=0.0 Nibs=1
                              + F6385 NZ%CAT(004F4) Type=0.0 Nibs=1
Read0  =  00000 NZ%SYM        -
Read1  =  00001 NZ%SYM        - F45B0 NZ%CAS(0031D) Type=0.0 Nibs=1
ReadD1 =  F0F8A NZ%GPR        -
ResetC =  00008 TI%R6S        -
Rewind =  00007 NZ%SYM        - F0BAB NZ%GPR(003E9) Type=0.0 Nibs=1
                              + F4586 NZ%CAS(002F3) Type=0.0 Nibs=1

S-R0-0 =  2F871 TI%R6S        -
S-R0-1 =  2F876 TI%R6S        -
S-R0-2 =  2F87B TI%R6S        -
S-R0-3 =  2F880 TI%R6S        - F21FA SC%ENT(002C6) Type=0.0 Nibs=5
S-R1-0 =  2F881 TI%R6S        -
S-R1-1 =  2F886 TI%R6S        - F2499 SC%ENT(00565) Type=0.0 Nibs=5
S-R1-2 =  2F88B TI%R6S        -
S-R1-3 =  2F890 TI%R6S        -
SALLOC =  0153B TI%R6S        -
SAVE1A =  F334D NZ%BIF        - F14A2 NZ%BAS(00508) Type=1.1 Nibs=4 Dist=01EAB
SAVE2C =  F3361 NZ%BIF        - F1683 NZ%BAS(006E9) Type=1.0 Nibs=4 Dist=01CDE
                              + F2CEB NZ%UTL(00055) Type=1.1 Nibs=3 Dist=00676
                              + F2DE8 NZ%UTL(00152) Type=1.1 Nibs=3 Dist=00579
                              + F73E0 NZ%FXQ(005C7) Type=1.0 Nibs=4 Dist=0407F
SAVED0 =  F32CD NZ%BIF        - F1170 NZ%BAS(001D6) Type=1.1 Nibs=4 Dist=0215D
                              + F2949 SC%ENT(00A15) Type=1.1 Nibs=4 Dist=00984
                              + F2CA5 NZ%UTL(0000F) Type=1.1 Nibs=3 Dist=00628
SAVED1 =  F32E3 NZ%BIF        - F1492 NZ%BAS(004F8) Type=1.1 Nibs=4 Dist=01E51
                              + F26BE SC%ENT(0078A) Type=1.1 Nibs=4 Dist=00C25
                              + F5EAB NZ%CAT(0001A) Type=1.1 Nibs=4 Dist=02BC8
                              + F5F2A NZ%CAT(00099) Type=1.1 Nibs=4 Dist=02C47
SAVEIT =  F3E4B NZ%BUT        - F1663 NZ%BAS(006C9) Type=1.0 Nibs=4 Dist=027E8
                              + F1F84 SC%ENT(00050) Type=1.1 Nibs=4 Dist=01EC7
                              + F1F97 SC%ENT(00063) Type=1.1 Nibs=4 Dist=01EB4
                              + F361A NZ%BIF(00743) Type=1.1 Nibs=4 Dist=00831
```

```
SAVESB =  0D66E TI%R6S        -
SAVEST =  F329C NZ%BIF        - F4103 NZ%BUT(0050C) Type=1.1 Nibs=4 Dist=00E67
SAVEXM =  0D663 TI%R6S        -
SAVGSB =  0D64E TI%R6S        -
SAVST+ =  F3463 NZ%BIF        -
SAVSTK =  2F59E TI%R6S        - F4256 NZ%BUT(0065F) Type=0.0 Nibs=5
SAVSTS =  F345A NZ%BIF        - F2ADF SC%ENT(00BAB) Type=1.1 Nibs=4 Dist=0097B
SB15S  =  0E19A TI%R6S        -
SCAN   =  04C40 TI%R6S        -
SCNRT  =  022B9 TI%R6S        - F3888 NZ%DSP(00251) Type=0.1 Nibs=5
                              + F39E1 NZ%DSP(003AA) Type=0.1 Nibs=5
SCOPCK =  0915B TI%R6S        -
SCREX0 =  2F941 TI%R6S        -
SCREX1 =  2F951 TI%R6S        -
SCREX2 =  2F961 TI%R6S        -
SCREX3 =  2F971 TI%R6S        -
SCRLLR =  0212E TI%R6S        - F5F9D NZ%CAT(0010C) Type=0.1 Nibs=5
SCROLT =  2F946 TI%R6S        -
SCRPTR =  2F966 TI%R6S        -
SCRST0 =  2F901 TI%R6S        -
SCRTCH =  2F901 TI%R6S        - F477F NZ%CAS(004EC) Type=0.0 Nibs=2
                              + F4979 NZ%CAS(006E6) Type=0.0 Nibs=4 Offset=     16
                              + F4A2E NZ%CAS(0079B) Type=0.0 Nibs=5
                              + F4B4D NZ%CAS(008BA) Type=0.0 Nibs=2 Offset=     20
                              + F4BBD NZ%CAS(0092A) Type=0.0 Nibs=2 Offset=     20
                              + F4BEA NZ%CAS(00957) Type=0.0 Nibs=5 Offset=     28
                              + F4C22 NZ%CAS(0098F) Type=0.0 Nibs=2 Offset=     36
                              + F4C5E NZ%CAS(009CB) Type=0.0 Nibs=2 Offset=     56
                              + F4C6D NZ%CAS(009DA) Type=0.0 Nibs=2 Offset=     20
                              + F4C98 NZ%CAS(00A05) Type=0.0 Nibs=5 Offset=     56
                              + F4CE7 NZ%CAS(00A54) Type=0.0 Nibs=2 Offset=     28
                              + F4D27 NZ%CAS(00A94) Type=0.0 Nibs=2 Offset=     28
                              + F4F32 NZ%CAS(00C9F) Type=0.0 Nibs=5 Offset=     16
                              + F4FF4 NZ%CAS(00D61) Type=0.0 Nibs=5 Offset=     36
                              + F55A7 NZ%HND(0048C) Type=0.0 Nibs=2 Offset=     56
                              + F581D NZ%HND(00702) Type=0.0 Nibs=5 Offset=     56
                              + F5845 NZ%HND(0072A) Type=0.0 Nibs=2 Offset=     32
                              + F5C3A NZ%HND(00B1F) Type=0.0 Nibs=5
                              + F5C95 NZ%HND(00B7A) Type=0.0 Nibs=5 Offset=     20
                              + F6360 NZ%CAT(004CF) Type=0.0 Nibs=5 Offset=     20
                              + F63A1 NZ%CAT(00510) Type=0.0 Nibs=5
                              + F6498 NZ%CAT(00607) Type=0.0 Nibs=5 Offset=     56
                              + F64D5 NZ%CAT(00644) Type=0.0 Nibs=5 Offset=     32
                              + F6589 NZ%CAT(006F8) Type=0.0 Nibs=5 Offset=     40
SE1-10 =  04468 TI%R6S        -
SECHMS =  13252 TI%R6S        -
SEEKA  =  F42C7 NZ%CAS        - F0B4E NZ%GPR(0038C) Type=1.1 Nibs=4 Dist=03779
                              + F12DD NZ%BAS(00343) Type=1.1 Nibs=4 Dist=02FEA
                              + F5C30 NZ%HND(00B15) Type=1.0 Nibs=4 Dist=01969
SEEKB  =  F42CE NZ%CAS        -
SEEKRD =  F636D NZ%CAT        -
SEND   =  F2CA0 NZ%UTL        - F0155 NZ%TBL(0014D) Type=1.2 Nibs=5 Dist=02B4B
SEND2O =  17DFA TI%R6S        - F6640 NZ%CAT(007AF) Type=0.1 Nibs=5
SENDEL =  17DC1 TI%R6S        -
SENDI+ =  F6A1E NZ%IOR        - F38CF NZ%DSP(00298) Type=1.1 Nibs=4 Dist=0314F
                              + F3B70 NZ%DSP(00539) Type=1.1 Nibs=4 Dist=02EAE
SENDIT =  F6A24 NZ%IOR        - F38E5 NZ%DSP(002AE) Type=1.1 Nibs=4 Dist=0313F
                              + F3AEC NZ%DSP(004B5) Type=1.1 Nibs=4 Dist=02F38
                              + F4322 NZ%CAS(0008F) Type=1.0 Nibs=4 Dist=02702
SENDWD =  17E15 TI%R6S        -
```

```
SENDd  =  F7D29 NZ%DEC        - F2C96 NZ%UTL(00000) Type=1.2 Nibs=5 Dist=05093
SENDp  =  F76C8 NZ%PAR        - F2C9B NZ%UTL(00005) Type=1.2 Nibs=5 Dist=04A2D
SETALM =  1290D TI%R6S        -
SETALR =  12917 TI%R6S        -
SETFMT =  0F01F TI%R6S        -
SETLP  =  F3C12 NZ%BUT        - F087F NZ%GPR(000BD) Type=1.1 Nibs=4 Dist=03393
                              + F367A NZ%DSP(00043) Type=1.1 Nibs=3 Dist=00598
SETSB  =  0D641 TI%R6S        -
SETTM0 =  13158 TI%R6S        -
SETTSR =  0FD01 NZ%SYM        -
SETUP  =  F3DC8 NZ%BUT        - F6E9C NZ%FXQ(00083) Type=1.0 Nibs=4 Dist=030D4
SFLAG?  =  1364C TI%R6S       - F098B NZ%GPR(001C9) Type=0.1 Nibs=5
SFLAGC =  13601 TI%R6S        -
SFLAGS =  135FA TI%R6S        -
SFLAGT =  13608 TI%R6S        -
SHF10  =  0C486 TI%R6S        -
SHFLAC =  0DB46 TI%R6S        -
SHFRAC =  0DB51 TI%R6S        -
SHFRBD =  0DB5F TI%R6S        -
SHRT   =  0F96C TI%R6S        -
SIGCHK =  0BD98 TI%R6S        -
SIGTST =  0E636 TI%R6S        -
SIN12  =  0D716 TI%R6S        -
SIN15  =  0D71A TI%R6S        -
SKIP   =  F7B36 NZ%PAR        -
SKIPDC =  057F6 TI%R6S        -
SLEEP  =  006C2 TI%R6S        -
SNAPBF =  2F7F0 TI%R6S        - F345E NZ%BIF(00587) Type=0.0 Nibs=5
                              + F3492 NZ%BIF(005BB) Type=0.0 Nibs=5 Offset=     33
SNAPR* =  01578 TI%R6S        -
SNAPRS =  01571 TI%R6S        - F52F7 NZ%HND(001DC) Type=0.1 Nibs=5
SNAPSV =  015A7 TI%R6S        -
SNDWD+ =  17E1F TI%R6S        -
SPACE  =  0AD9D TI%R6S        -
SPLITA =  0C6BF TI%R6S        -
SPLITC =  0C940 TI%R6S        -
SPLTAC =  0C934 TI%R6S        -
SPLTAX =  0E62B TI%R6S        -
SPOLL  =  F1B9D NZ%BAS        - F00E0 NZ%TBL(000D8) Type=1.2 Nibs=5 Dist=01ABD
SQR15  =  0C534 TI%R6S        -
SQR17  =  0C553 TI%R6S        -
SQR70  =  0C5C3 TI%R6S        -
SQRSAV =  0D629 TI%R6S        -
SRLEAS =  015EC TI%R6S        -
ST!NOd =  F7D83 NZ%DEC        -
ST!NOp =  F782C NZ%PAR        -
STAB1  =  0D3D9 TI%R6S        -
STAB2  =  0D400 TI%R6S        -
STANBY =  F16AF NZ%BAS        - F01C1 NZ%TBL(001B9) Type=1.2 Nibs=5 Dist=014EE
STANDd =  F7C74 NZ%DEC        - F16A5 NZ%BAS(0070B) Type=1.2 Nibs=5 Dist=065CF
STANDp =  F75CD NZ%PAR        - F16AA NZ%BAS(00710) Type=1.2 Nibs=5 Dist=05F23
STANd+ =  F7C6C NZ%DEC        -
STANp+ =  F75BC NZ%PAR        -
START  =  F087D NZ%GPR        - F1029 NZ%BAS(0008F) Type=1.1 Nibs=3 Dist=007AC
                              + F14EA NZ%BAS(00550) Type=1.1 Nibs=4 Dist=00C6D
                              + F1D11 NZ%BAS(00D77) Type=1.1 Nibs=4 Dist=01494
                              + F22B6 SC%ENT(00382) Type=1.1 Nibs=4 Dist=01A39
                              + F2324 SC%ENT(003F0) Type=1.1 Nibs=4 Dist=01AA7
                              + F2A1D SC%ENT(00AE9) Type=1.1 Nibs=4 Dist=021A0
                              + F3691 NZ%DSP(0005A) Type=1.1 Nibs=4 Dist=02E14
```

```
                                + F4A85 NZ%CAS(007F2) Type=1.0 Nibs=4 Dist=04208
                                + F542D NZ%HND(00312) Type=1.0 Nibs=4 Dist=04BB0
                                + F5F47 NZ%CAT(000B6) Type=1.1 Nibs=4 Dist=056CA
                                + F5FB6 NZ%CAT(00125) Type=1.1 Nibs=4 Dist=05739
                                + F6E93 NZ%FXQ(0007A) Type=1.1 Nibs=4 Dist=06616
START+ =  F0883 NZ%GPR          - F2CAC NZ%UTL(00016) Type=1.1 Nibs=4 Dist=02429
START- =  F0886 NZ%GPR          - F19AE NZ%BAS(00A14) Type=1.1 Nibs=4 Dist=01128
                                + F1A6E NZ%BAS(00AD4) Type=1.1 Nibs=4 Dist=011E8
STATAR =  2F7AD TI%R6S          -
STATRS =  172F3 TI%R6S          -
STATSV =  1732F TI%R6S          -
STATUS =  F1DEF NZ%BAS          - F00FB NZ%TBL(000F3) Type=1.2 Nibs=5 Dist=01CF4
STCD2  =  0D427 TI%R6S          -
STKCHR =  18504 TI%R6S          -
STKCMD =  155ED TI%R6S          -
STKVCT =  1470C TI%R6S          - F2258 SC%ENT(00324) Type=0.1 Nibs=5
STMBCL =  090E7 TI%R6S          -
STMBUF =  090DF TI%R6S          -
STMTD0 =  2F891 TI%R6S          - F15D8 NZ%BAS(0063E) Type=0.0 Nibs=2 Offset=      2
                                + F1692 NZ%BAS(006F8) Type=0.0 Nibs=5
                                + F186D NZ%BAS(008D3) Type=0.0 Nibs=2
                                + F26AE SC%ENT(0077A) Type=0.0 Nibs=5
                                + F282A SC%ENT(008F6) Type=0.0 Nibs=5
                                + F32D4 NZ%BIF(003FD) Type=0.0 Nibs=5
                                + F3300 NZ%BIF(00429) Type=0.0 Nibs=5
                                + F3328 NZ%BIF(00451) Type=0.0 Nibs=5
                                + F333F NZ%BIF(00468) Type=0.0 Nibs=5
STMTD1 =  2F896 TI%R6S          - F2850 SC%ENT(0091C) Type=0.0 Nibs=5
                                + F28C4 SC%ENT(00990) Type=0.0 Nibs=5
                                + F32EA NZ%BIF(00413) Type=0.0 Nibs=5
                                + F3313 NZ%BIF(0043C) Type=0.0 Nibs=5
                                + F73CD NZ%FXQ(005B4) Type=0.0 Nibs=5
STMTR0 =  2F871 TI%R6S          - F1118 NZ%BAS(0017E) Type=0.0 Nibs=4 Offset=     11
                                + F1689 NZ%BAS(006EF) Type=0.0 Nibs=5
                                + F3354 NZ%BIF(0047D) Type=0.0 Nibs=5
                                + F3385 NZ%BIF(004AE) Type=0.0 Nibs=5
                                + F54A2 NZ%HND(00387) Type=0.0 Nibs=5 Offset=      1
STMTR1 =  2F881 TI%R6S          - F10C4 NZ%BAS(0012A) Type=0.0 Nibs=2 Offset=      2
                                + F10F7 NZ%BAS(0015D) Type=0.0 Nibs=5 Offset=      2
                                + F336B NZ%BIF(00494) Type=0.0 Nibs=5
                                + F3399 NZ%BIF(004C2) Type=0.0 Nibs=5
                                + F51E8 NZ%HND(000CD) Type=0.0 Nibs=5 Offset=      5
                                + F5453 NZ%HND(00338) Type=0.0 Nibs=5 Offset=      2
                                + F5499 NZ%HND(0037E) Type=0.0 Nibs=5 Offset=      9
                                + F59BE NZ%HND(008A3) Type=0.0 Nibs=5 Offset=     14
                                + F5A3D NZ%HND(00922) Type=0.0 Nibs=5 Offset=     14
STORE  =  0F5F8 TI%R6S          - F218B SC%ENT(00257) Type=0.1 Nibs=5
STR$00 =  1815C TI%R6S          -
STR$SB =  18149 TI%R6S          -
STRASN =  0F6B3 TI%R6S          -
STREQL =  1B1EF TI%R6S          -
STRGCK =  036BA TI%R6S          -
STRHDR =  0F09A TI%R6S          -
STRHED =  14C2E TI%R6S          - F264B SC%ENT(00717) Type=0.1 Nibs=5
STRNGP =  0379D TI%R6S          -
STRTST =  1B1C7 TI%R6S          -
STSAVE =  2F6BE TI%R6S          - F32A3 NZ%BIF(003CC) Type=0.0 Nibs=5
                                + F32BE NZ%BIF(003E7) Type=0.0 Nibs=5
STSCR  =  0E92C TI%R6S          -
STUFF  =  1B0B2 TI%R6S          -
```

```
SUBONE =  0C327 TI%R6S        -
SVDOD1 =  F7A93 NZ%PAR        -
SVINF+ =  08457 TI%R6S        -
SVINFO =  0845A TI%R6S        -
SVTRC  =  0FA35 TI%R6S        - F22D1 SC%ENT(0039D) Type=0.1 Nibs=5
SWAPO1 =  F65FB NZ%CAT        - F093B NZ%GPR(00179) Type=1.1 Nibs=4 Dist=05CC0
                              + F094C NZ%GPR(0018A) Type=1.1 Nibs=4 Dist=05CAF
                              + F1185 NZ%BAS(001EB) Type=1.1 Nibs=4 Dist=05476
SWAPDO =  F331F NZ%BIF        - F2956 SC%ENT(00A22) Type=1.1 Nibs=4 Dist=009C9
                              + F2D05 NZ%UTL(0006F) Type=1.1 Nibs=3 Dist=0061A
                              + F2D42 NZ%UTL(000AC) Type=1.1 Nibs=3 Dist=005DD
                              + F2D77 NZ%UTL(000E1) Type=1.1 Nibs=3 Dist=005A8
SWPBYT =  17A24 TI%R6S        -
SYNTXe =  02E2B TI%R6S        -
SYSEN  =  2F58A TI%R6S        -
SYSFLG =  2F6D9 TI%R6S        -
SavLvl =  00005 TI%R6S        -
Seek   =  00004 NZ%SYM        - F42CF NZ%CAS(0003C) Type=0.0 Nibs=1
Seeka  =  F5C2E NZ%HND        - F637E NZ%CAT(004ED) Type=1.1 Nibs=3 Dist=00750
SendBf =  F39DF NZ%DSP        -
SetAVM =  1B9FA TI%R6S        -
SetBP  =  00003 NZ%SYM        - F130B NZ%BAS(00371) Type=0.0 Nibs=1
                              + F4896 NZ%CAS(00603) Type=0.0 Nibs=1
                              + F4EB2 NZ%CAS(00C1F) Type=0.0 Nibs=1
                              + F502B NZ%CAS(00D98) Type=0.0 Nibs=1
SngDev =  00004 NZ%SYM        - F3D4A NZ%BUT(00153) Type=0.0 Nibs=1
                              + F3E61 NZ%BUT(0026A) Type=0.0 Nibs=1
                              + F3EDF NZ%BUT(002E8) Type=0.0 Nibs=1
SpChar =  00002 NZ%PAR        -
StarOK =  0000A NZ%PAR        - F7623 NZ%PAR(00126) Type=0.0 Nibs=1
                              + F78B4 NZ%PAR(003B7) Type=0.0 Nibs=1
                              + F78CA NZ%PAR(003CD) Type=0.0 Nibs=1
                              + F79B2 NZ%PAR(004B5) Type=0.0 Nibs=1
                              + F79B9 NZ%PAR(004BC) Type=0.0 Nibs=1
                              + F7B8B NZ%PAR(0068E) Type=0.0 Nibs=1
StrOK  =  0000A NZ%PAR        - F7739 NZ%PAR(0023C) Type=0.0 Nibs=1
                              + F77DD NZ%PAR(002E0) Type=0.0 Nibs=1
                              + F7811 NZ%PAR(00314) Type=0.0 Nibs=1
                              + F7848 NZ%PAR(0034B) Type=0.0 Nibs=1
                              + F7B63 NZ%PAR(00666) Type=0.0 Nibs=1

TALK   =  F0D44 NZ%GPR        - F2A38 SC%ENT(00B04) Type=1.1 Nibs=4 Dist=01CF4
TAN12  =  0D72F TI%R6S        -
TAN15  =  0D733 TI%R6S        -
TASTK  =  2F599 TI%R6S        -
TBLJMC =  02426 TI%R6S        -
TBLJMP =  0242A TI%R6S        -
TBMSG$ =  099AB TI%R6S        -
TER/LF =  F24FD SC%ENT        - F5BC9 NZ%HND(00AAE) Type=1.0 Nibs=4 Dist=036CC
TERCHR =  2F97D TI%R6S        - F22EF SC%ENT(003BB) Type=0.0 Nibs=5
                              + F27F8 SC%ENT(008C4) Type=0.0 Nibs=5
                              + F2F57 NZ%BIF(00080) Type=0.0 Nibs=4
TFHDLR =  1702F TI%R6S        -
TFORN  =  2F59E TI%R6S        -
TGSBS  =  2F5A3 TI%R6S        -
TIMAF  =  2F787 TI%R6S        -
TIMER1 =  2E3F8 TI%R6S        -
TIMER2 =  2E2F8 TI%R6S        -
TIMER3 =  2E1F8 TI%R6S        -
TIMLAF =  2F77B TI%R6S        -
```

```
TIMLST = 2F76F TI%R6S       -
TIMOFS = 2F763 TI%R6S       -
TKSCN+ = 08A6B TI%R6S       -
TKSCN7 = 08A99 TI%R6S       -
TMRAD1 = 2F697 TI%R6S       -
TMRAD2 = 2F69C TI%R6S       -
TMRAD3 = 2F6A1 TI%R6S       -
TMRIN1 = 2F6A6 TI%R6S       -
TMRIN2 = 2F6AE TI%R6S       -
TMRIN3 = 2F6B6 TI%R6S       -
TODT   = 13229 TI%R6S       -
TONE   = 0EBEB TI%R6S       -
TRACDC = 052FC TI%R6S       -
TRACEM = 2F7B0 TI%R6S       -
TRC90  = 0DA11 TI%R6S       -
TRES2C = F3446 NZ%BIF       - F5476 NZ%HND(0035B) Type=1.1 Nibs=4 Dist=02030
                            + F55C6 NZ%HND(004AB) Type=1.1 Nibs=4 Dist=02180
                            + F5B54 NZ%HND(00A39) Type=1.1 Nibs=4 Dist=0270E
                            + F6211 NZ%CAT(00380) Type=1.1 Nibs=4 Dist=02DCB
TRESD0 = F33D2 NZ%BIF       - F16A1 NZ%BAS(00707) Type=1.0 Nibs=4 Dist=01D31
                            + F1F9D SC%ENT(00069) Type=1.1 Nibs=4 Dist=01435
                            + F65F5 NZ%CAT(00764) Type=1.1 Nibs=4 Dist=03223
                            + F74BE NZ%FXQ(006A5) Type=1.1 Nibs=4 Dist=040EC
                            + F74ED NZ%FXQ(006D4) Type=1.1 Nibs=4 Dist=0411B
TRESD1 = F33E5 NZ%BIF       - F0FE6 NZ%BAS(0004C) Type=1.0 Nibs=4 Dist=023FF
                            + F214C SC%ENT(00218) Type=1.1 Nibs=4 Dist=01299
TRFMBF = 2F8C5 TI%R6S       -
TRFROM = 0FE59 TI%R6S       -
TRIGER = F155B NZ%BAS       - F01A6 NZ%TBL(0019E) Type=1.2 Nibs=5 Dist=013B5
TRIGd  = F7CC7 NZ%DEC       - F1551 NZ%BAS(005B7) Type=1.2 Nibs=5 Dist=06776
TRIGp  = F761E NZ%PAR       - F1556 NZ%BAS(005BC) Type=1.2 Nibs=5 Dist=060C8
TRKDON = 1CFAC TI%R6S       -
TRMNTR = 0F1DD TI%R6S       -
TRPREG = 2F6F9 TI%R6S       -
TRSFMu = 16B84 TI%R6S       -
TRTO+  = 0FE7B TI%R6S       -
TSAV2C = F3429 NZ%BIF       - F546A NZ%HND(0034F) Type=1.1 Nibs=4 Dist=02041
                            + F55B4 NZ%HND(00499) Type=1.1 Nibs=4 Dist=0218B
                            + F5B1D NZ%HND(00A02) Type=1.1 Nibs=4 Dist=026F4
                            + F61FE NZ%CAT(0036D) Type=1.1 Nibs=4 Dist=02DD5
TSAVD0 = F33A6 NZ%BIF       - F169B NZ%BAS(00701) Type=1.0 Nibs=4 Dist=01D0B
                            + F639B NZ%CAT(0050A) Type=1.1 Nibs=4 Dist=02FF5
                            + F6E85 NZ%FXQ(0006C) Type=1.1 Nibs=4 Dist=03ADF
                            + F74A1 NZ%FXQ(00688) Type=1.1 Nibs=4 Dist=040FB
                            + F74E0 NZ%FXQ(006C7) Type=1.1 Nibs=4 Dist=0413A
TSAVD1 = F33BC NZ%BIF       - F0FA3 NZ%BAS(00009) Type=1.1 Nibs=4 Dist=02419
                            + F19DE NZ%BAS(00A44) Type=1.1 Nibs=4 Dist=019DE
                            + F1CDA NZ%BAS(00D40) Type=1.1 Nibs=4 Dist=016E2
                            + F2138 SC%ENT(00204) Type=1.1 Nibs=4 Dist=01284
                            + F544D NZ%HND(00332) Type=1.1 Nibs=4 Dist=02091
TST12A = 0D476 TI%R6S       -
TST15  = 0D47A TI%R6S       -
TSTAT  = F4293 NZ%CAS       - F0B2E NZ%GPR(0036C) Type=1.1 Nibs=4 Dist=03765
                            + F1657 NZ%BAS(006BD) Type=1.0 Nibs=4 Dist=02C3C
                            + F5C7E NZ%HND(00B63) Type=1.1 Nibs=4 Dist=019EB
                            + F6376 NZ%CAT(004E5) Type=1.1 Nibs=4 Dist=020E3
TSTATA = F429A NZ%CAS       - F0B69 NZ%GPR(003A7) Type=1.1 Nibs=4 Dist=03731
                            + F6391 NZ%CAT(00500) Type=1.0 Nibs=4 Dist=020F7
TSWAD1 = F33F8 NZ%BIF       - F1A8F NZ%BAS(00AF5) Type=1.1 Nibs=4 Dist=01969
                            + F1AE5 NZ%BAS(00B4B) Type=1.1 Nibs=4 Dist=01913
```

```
                                + F1AF8 NZ%BAS(00B5E) Type=1.1 Nibs=4 Dist=01900
TWO*    =  0DB38 TI%R6S         -
Timout  =  007D0 NZ%SYM         - F16D5 NZ%BAS(0073B) Type=0.0 Nibs=5
Trace   =  0000F TI%R6S         -
TstEnd  =  1C0FF TI%R6S         - F2681 SC%ENT(0074D) Type=0.1 Nibs=5

UCRANG  =  F0E66 NZ%GPR         - F1AA5 NZ%BAS(00B0B) Type=1.1 Nibs=4 Dist=00C3F
                                + F1ABE NZ%BAS(00B24) Type=1.1 Nibs=4 Dist=00C58
                                + F74F9 NZ%FXQ(006E0) Type=1.0 Nibs=4 Dist=06693
ULYL    =  F0CEA NZ%GPR         - F103A NZ%BAS(000A0) Type=1.1 Nibs=3 Dist=00350
                                + F12EB NZ%BAS(00351) Type=1.1 Nibs=3 Dist=00601
UNFNIB  =  2F6FA TI%R6S         -
UNLPUT  =  F0D00 NZ%GPR         - F15E5 NZ%BAS(0064B) Type=1.1 Nibs=4 Dist=008E5
                                + F2A2F SC%ENT(00AFB) Type=1.1 Nibs=4 Dist=01D2F
UNP     =  00001 TI%R6S         -
UNT     =  F24E6 SC%ENT         - F0867 NZ%GPR(000A5) Type=1.1 Nibs=4 Dist=01C7F
                                + F0AAE NZ%GPR(002EC) Type=1.1 Nibs=4 Dist=01A38
UPCPOS  =  13C67 TI%R6S         -
UPD1EN  =  2F599 TI%R6S         -
UPD1ST  =  2F55D TI%R6S         -
UPD2EN  =  2F6A6 TI%R6S         -
UPD2ST  =  2F674 TI%R6S         -
UPDANN  =  13571 TI%R6S         -
USGch+  =  1BC15 TI%R6S         -
USGch-  =  1BC0B TI%R6S         -
USGrst  =  1BC63 TI%R6S         - F27A0 SC%ENT(0086C) Type=0.1 Nibs=5
USING   =  1B446 TI%R6S         - F1FBE SC%ENT(0008A) Type=0.1 Nibs=5
USINGp  =  03628 TI%R6S         - F7527 NZ%PAR(0002A) Type=0.1 Nibs=5
USloop  =  1C14B TI%R6S         - F2790 SC%ENT(0085C) Type=0.1 Nibs=5
USnm05  =  1BD12 TI%R6S         -
USst03  =  1BBCE TI%R6S         -
USst05  =  1BBD4 TI%R6S         -
UTLEND  =  F0861 NZ%GPR         - F10DC NZ%BAS(00142) Type=1.1 Nibs=4 Dist=0087B
                                + F4590 NZ%CAS(002FD) Type=1.0 Nibs=4 Dist=03D2F
                                + F4E69 NZ%CAS(00BD6) Type=1.0 Nibs=4 Dist=04608
                                + F5AD3 NZ%HND(009B8) Type=1.1 Nibs=4 Dist=05272
                                + F5F6F NZ%CAT(000DE) Type=1.1 Nibs=4 Dist=0570E
                                + F5F83 NZ%CAT(000F2) Type=1.1 Nibs=4 Dist=05722
Ucrang  =  F74F7 NZ%FXQ         - F776A NZ%PAR(0026D) Type=1.1 Nibs=3 Dist=00273
                                + F7793 NZ%PAR(00296) Type=1.1 Nibs=3 Dist=0029C
Utlend  =  F4E67 NZ%CAS         - F52EB NZ%HND(001D0) Type=1.1 Nibs=3 Dist=00484

VALOO   =  1AD8F TI%R6S         -
VALCHK  =  1AE61 TI%R6S         -
VARDC   =  0537C TI%R6S         -
VARNB-  =  0E28D TI%R6S         -
VARNBR  =  0E289 TI%R6S         -
VARP    =  0350E TI%R6S         -
VECTOR  =  2F43C TI%R6S         -
VIEWD1  =  15147 TI%R6S         -
VRIABL  =  04BC4 TI%R6S         -
ValSub  =  0000A TI%R6S         -
Verify  =  0000B NZ%SYM         -
VolLbl  =  0005F NZ%SYM         - F0AE3 NZ%GPR(00321) Type=0.0 Nibs=2
                                + F3D8F NZ%BUT(00198) Type=0.0 Nibs=2
                                + F700E NZ%FXQ(001F5) Type=0.0 Nibs=2
                                + F7311 NZ%FXQ(004F8) Type=0.0 Nibs=2

WFTMDT  =  085DD TI%R6S         -
WINDLN  =  2F473 TI%R6S         -
```

```
WINDST =  2F471 TI%R6S        -
WIPOUT =  1B0AF TI%R6S        -
WRBYTC =  13A73 TI%R6S        -
WRDSC+ =  02C26 TI%R6S        -
WRDSCN =  02C2A TI%R6S        - F7B95 NZ%PAR(00698) Type=0.1 Nibs=5
WRITE# =  F45D4 NZ%CAS        - F5327 NZ%HND(0020C) Type=1.1 Nibs=4 Dist=00D53
                              + F534C NZ%HND(00231) Type=1.1 Nibs=4 Dist=00D78
WRITIT =  F69AF NZ%IOR        - F109A NZ%BAS(00100) Type=1.1 Nibs=4 Dist=05915
                              + F37E6 NZ%DSP(001AF) Type=1.1 Nibs=4 Dist=031C9
                              + F3A85 NZ%DSP(0044E) Type=1.1 Nibs=4 Dist=02F2A
                              + F4A63 NZ%CAS(007D0) Type=1.0 Nibs=4 Dist=01F4C
WRITNB =  1752B TI%R6S        -
WRTASC =  F6653 NZ%CAT        - F1820 NZ%BAS(00886) Type=1.1 Nibs=4 Dist=04E33
                              + F18CA NZ%BAS(00930) Type=1.1 Nibs=4 Dist=04D89
WRTFIB =  11CEE TI%R6S        -
WRTNUM =  139C4 TI%R6S        -
WRTSTR =  1396F TI%R6S        -
WSTRFX =  138B5 TI%R6S        -
Wallby =  0000A NZ%SYM        - F366E NZ%DSP(00037) Type=0.0 Nibs=1
                              + F36A8 NZ%DSP(00071) Type=0.0 Nibs=1
                              + F36D1 NZ%DSP(0009A) Type=0.0 Nibs=1
                              + F3A1F NZ%DSP(003E8) Type=0.0 Nibs=1
                              + F3AA3 NZ%DSP(0046C) Type=0.0 Nibs=1
Write  =  00002 NZ%SYM        - F4A36 NZ%CAS(007A3) Type=0.0 Nibs=1
                              + F5C8C NZ%HND(00B71) Type=0.0 Nibs=1
Write0 =  00000 NZ%SYM        - F5045 NZ%CAS(00DB2) Type=0.0 Nibs=1
Write1 =  00001 NZ%SYM        - F1326 NZ%BAS(0038C) Type=0.0 Nibs=1

XDelay =  00009 TI%R6S        -
XMTADR =  08133 TI%R6S        -
XROM01 =  00001 TI%R6S        -
XWORDd =  F7C5B NZ%DEC        -
XWORDp =  F79AE NZ%PAR        -
XWRD1p =  F75B1 NZ%PAR        -
XXHEAD =  1A44E TI%R6S        -
XYEX   =  0C697 TI%R6S        -
XchgL  =  0000A NZ%SYM        -
XchgT  =  00004 NZ%SYM        - F12E5 NZ%BAS(0034B) Type=0.0 Nibs=1
                              + F12FC NZ%BAS(00362) Type=0.0 Nibs=1
                              + F45A7 NZ%CAS(00314) Type=0.0 Nibs=1
                              + F45C8 NZ%CAS(00335) Type=0.0 Nibs=1
XfrO1L =  00009 NZ%SYM        -
XfrO1T =  00005 NZ%SYM        -

YMDDAY =  13304 TI%R6S        -
YMDHO1 =  130E5 TI%R6S        -
YMDHMS =  130DB TI%R6S        - F4AAC NZ%CAS(00819) Type=0.1 Nibs=5
YTML   =  F0D30 NZ%GPR        - F1BAF NZ%BAS(00C15) Type=1.1 Nibs=4 Dist=00E7F
                              + F2396 SC%ENT(00462) Type=1.1 Nibs=4 Dist=01666
                              + F4846 NZ%CAS(005B3) Type=1.0 Nibs=4 Dist=03B16
                              + F6891 NZ%IOR(001BF) Type=1.1 Nibs=4 Dist=05B61
YTMLL  =  F0D37 NZ%GPR        - F68A7 NZ%IOR(001D5) Type=1.1 Nibs=4 Dist=05B70
YX2-12 =  0D274 TI%R6S        -
YX2-15 =  0D27A TI%R6S        -

ZERBUF =  18B20 TI%R6S        -

a!     =  00021 TI%R6S        -
a"     =  00022 TI%R6S        -
a$     =  00024 TI%R6S        -
```

```
a'      =  00027 TI%R6S        -
a.      =  0002E TI%R6S        -
a0      =  00030 TI%R6S        -
a1      =  00031 TI%R6S        -
a2      =  00032 TI%R6S        -
a3      =  00033 TI%R6S        -
a4      =  00034 TI%R6S        -
a5      =  00035 TI%R6S        -
a6      =  00036 TI%R6S        -
a7      =  00037 TI%R6S        -
a8      =  00038 TI%R6S        -
a9      =  00039 TI%R6S        -
aVE=D1  =  F21BB SC%ENT        - F60FD NZ%CAT(0026C) Type=1.1 Nibs=4 Dist=03F42
                               + F61AA NZ%CAT(00319) Type=1.1 Nibs=4 Dist=03FEF

bALTCH  =  00BFB TI%R6S        -
bASSGN  =  00804 TI%R6S        -
bCARD   =  00807 TI%R6S        -
bCHARS  =  00BFB TI%R6S        -
bECOMD  =  00809 TI%R6S        -
bFIB    =  00803 TI%R6S        - F4BF9 NZ%CAS(00966) Type=0.0 Nibs=3
                               + F5D2D NZ%HND(00C12) Type=0.0 Nibs=3
bFILE   =  00805 TI%R6S        -
bIEXKY  =  00802 TI%R6S        -
bLEX    =  00BFC TI%R6S        -
bPILAI  =  00810 TI%R6S        - F0941 NZ%GPR(0017F) Type=0.0 Nibs=3
                               + F17D5 NZ%BAS(0083B) Type=0.0 Nibs=3
                               + F1A23 NZ%BAS(00A89) Type=0.0 Nibs=3
                               + F1B2C NZ%BAS(00B92) Type=0.0 Nibs=3
                               + F2FD9 NZ%BIF(00102) Type=0.0 Nibs=3
                               + F4126 NZ%BUT(0052F) Type=0.0 Nibs=3
bPILSV  =  0080F TI%R6S        - F2EE0 NZ%BIF(00009) Type=0.0 Nibs=3
                               + F2F93 NZ%BIF(000BC) Type=0.0 Nibs=3
bROMTB  =  00BFE TI%R6S        -
bSCRTC  =  00E00 TI%R6S        -
bSERR   =  F1A30 NZ%BAS        - F34C7 NZ%BIF(005F0) Type=1.0 Nibs=4 Dist=01A97
bSTART  =  00808 TI%R6S        -
bSTAT   =  00806 TI%R6S        -
bSTMT   =  00801 TI%R6S        -
bSTMXQ  =  00811 TI%R6S        - F2C34 SC%ENT(00D00) Type=0.0 Nibs=3
                               + F2FD0 NZ%BIF(000F9) Type=0.0 Nibs=3

cATCH+  =  F7B14 NZ%PAR        - F1AD4 NZ%BAS(00B3A) Type=1.1 Nibs=4 Dist=06040
cC->C   =  00068 TI%R6S        -
cR->C   =  00069 TI%R6S        -
cRCL    =  00067 TI%R6S        -

dCARD   =  00007 TI%R6S        -
dIRAM   =  00001 TI%R6S        -
dMAIN   =  00000 TI%R6S        -
dPCRD   =  00007 TI%R6S        -
dPORT   =  00001 TI%R6S        -

e#of#   =  000F7 TI%R6S        -
e0^0    =  00006 TI%R6S        -
e0^NEG  =  00005 TI%R6S        -
e1^INF  =  00011 TI%R6S        -
e2MROM  =  0001A TI%R6S        -
eABORT  =  00034 NZ%ERR        - F0BEB NZ%GPR(00429) Type=0.0 Nibs=1
                               + F1604 NZ%BAS(0066A) Type=0.0 Nibs=1
```

```
                                    + F240C SC%ENT(004D8) Type=0.0 Nibs=1
                                    + F243A SC%ENT(00506) Type=0.0 Nibs=1
                                    + F317B NZ%BIF(002A4) Type=0.0 Nibs=1
                                    + F34F6 NZ%BIF(0061F) Type=0.0 Nibs=1
                                    + F350F NZ%BIF(00638) Type=0.0 Nibs=1
                                    + F3513 NZ%BIF(0063C) Type=0.0 Nibs=1
                                    + F6710 NZ%IOR(0003E) Type=0.0 Nibs=1
                                    + F67BF NZ%IOR(000ED) Type=0.0 Nibs=1
                                    + F6947 NZ%IOR(00275) Type=0.0 Nibs=1
                                    + F6B2A NZ%IOR(00458) Type=0.0 Nibs=1
eAF    =  0001B TI%R6S              -
eALGN  =  000F0 TI%R6S              -
eBADMD =  00029 NZ%ERR              - F08AD NZ%GPR(000EB) Type=0.0 Nibs=1
                                    + F0C75 NZ%GPR(004B3) Type=0.0 Nibs=1
                                    + F2908 SC%ENT(009D4) Type=0.0 Nibs=1
eBLANK =  00018 NZ%ERR              -
eCALGN =  00060 TI%R6S              -
eCHNL# =  00029 TI%R6S              -
eCHSUM =  0001A NZ%ERR              -
eDATTY =  0001F TI%R6S              - F061F NZ%ERR(00215) Type=0.0 Nibs=2
eDEVIC =  00041 NZ%ERR              - F0515 NZ%ERR(0010B) Type=0.0 Nibs=2
                                    + F05E6 NZ%ERR(001DC) Type=0.0 Nibs=2
                                    + F060E NZ%ERR(00204) Type=0.0 Nibs=2
eDIRFL =  0001F NZ%ERR              - F4DA4 NZ%CAS(00B11) Type=0.0 Nibs=1
eDSPEC =  00035 NZ%ERR              - F1193 NZ%BAS(001F9) Type=0.0 Nibs=1
                                    + F1B13 NZ%BAS(00B79) Type=0.0 Nibs=1
                                    + F1D1E NZ%BAS(00D84) Type=0.0 Nibs=1
                                    + F1F71 SC%ENT(0003D) Type=0.0 Nibs=1
                                    + F47B0 NZ%CAS(0051D) Type=0.0 Nibs=1
                                    + F60D5 NZ%CAT(00244) Type=0.0 Nibs=1
                                    + F6E7E NZ%FXQ(00065) Type=0.0 Nibs=1
                                    + F6F4D NZ%FXQ(00134) Type=0.0 Nibs=1
                                    + F7204 NZ%FXQ(003EB) Type=0.0 Nibs=1
                                    + F73B5 NZ%FXQ(0059C) Type=0.0 Nibs=1
                                    + F7428 NZ%FXQ(0060F) Type=0.0 Nibs=1
eDTYPE =  0002F NZ%ERR              - F1508 NZ%BAS(0056E) Type=0.0 Nibs=1
                                    + F35B0 NZ%BIF(006D9) Type=0.0 Nibs=1
                                    + F4309 NZ%CAS(00076) Type=0.0 Nibs=1
eDVCNF =  00040 TI%R6S              - F050D NZ%ERR(00103) Type=0.0 Nibs=2
eEFILE =  0001E NZ%ERR              - F4BB5 NZ%CAS(00922) Type=0.0 Nibs=1
                                    + F5D9A NZ%HND(00C7F) Type=0.0 Nibs=1
eEOFIL =  00036 TI%R6S              -
eEOTAP =  00011 NZ%ERR              - F4DAD NZ%CAS(00B1A) Type=0.0 Nibs=1
eEXCHR =  0004E TI%R6S              - F0438 NZ%ERR(0002E) Type=0.0 Nibs=2
eEXPO  =  00003 TI%R6S              -
eEXPCT =  000E7 TI%R6S              -
eF2BIG =  0004A TI%R6S              -
eFACCS =  0003C TI%R6S              -
eFEXST =  0003B TI%R6S              - F04E2 NZ%ERR(000D8) Type=0.0 Nibs=2
eFILE  =  000EA TI%R6S              - F04DA NZ%ERR(000D0) Type=0.0 Nibs=2
eFLOST =  00024 NZ%ERR              - F055F NZ%ERR(00155) Type=0.0 Nibs=2
                                    + F0567 NZ%ERR(0015D) Type=0.0 Nibs=2
                                    + F058E NZ%ERR(00184) Type=0.0 Nibs=2
eFNNtF =  00021 TI%R6S              -
eFOPEN =  0003E TI%R6S              -
eFPROT =  0003D TI%R6S              - F0460 NZ%ERR(00056) Type=0.0 Nibs=2
eFRAME =  00040 NZ%ERR              - F054C NZ%ERR(00142) Type=0.0 Nibs=2
                                    + F0586 NZ%ERR(0017C) Type=0.0 Nibs=2
eFRTOI =  0002A NZ%ERR              -
eFRTOL =  0002B NZ%ERR              -
```

```
eFSPEC =  0003A TI%R6S        -
eFTYPE =  0003F TI%R6S        - F57F3 NZ%HND(006D8) Type=0.0 Nibs=4
eFnFND =  00039 TI%R6S        - F04A1 NZ%ERR(00097) Type=0.0 Nibs=2
                              + F5634 NZ%HND(00519) Type=0.0 Nibs=4
eFwoNX =  0002A TI%R6S        -
eHPIL  =  00000 NZ%ERR        -
eIF*ZR =  00010 TI%R6S        -
eIF-IF =  0000F TI%R6S        -
eIF/IF =  0000E TI%R6S        -
eILCNT =  0004F TI%R6S        -
eILEXP =  00050 TI%R6S        - F0450 NZ%ERR(00046) Type=0.0 Nibs=2
eILEXp =  00006 NZ%ERR        - F7A81 NZ%PAR(00584) Type=0.0 Nibs=1
                              + F7B0A NZ%PAR(0060D) Type=0.0 Nibs=1
eILKEY =  00055 TI%R6S        -
eILLEG =  000E6 TI%R6S        -
eILPAR =  00051 TI%R6S        - F0448 NZ%ERR(0003E) Type=0.0 Nibs=2
eILPAr =  00005 NZ%ERR        - F76BD NZ%PAR(001C0) Type=0.0 Nibs=1
                              + F78C7 NZ%PAR(003CA) Type=0.0 Nibs=1
eILTFM =  00037 TI%R6S        -
eILVAR =  00053 TI%R6S        -
eIMGOV =  0002F TI%R6S        -
eINF   =  000F3 TI%R6S        -
eINF^0 =  00012 TI%R6S        -
eINPUT =  000F4 TI%R6S        -
eINVAL =  00012 NZ%ERR        - F048A NZ%ERR(00080) Type=0.0 Nibs=2
                              + F04A9 NZ%ERR(0009F) Type=0.0 Nibs=2
                              + F04B1 NZ%ERR(000A7) Type=0.0 Nibs=2
                              + F04B9 NZ%ERR(000AF) Type=0.0 Nibs=2
                              + F04C1 NZ%ERR(000B7) Type=0.0 Nibs=2
eINVIM =  0002D TI%R6S        -
eINVLD =  000EC TI%R6S        - F047F NZ%ERR(00075) Type=0.0 Nibs=2
                              + F0596 NZ%ERR(0018C) Type=0.0 Nibs=2
                              + F060B NZ%ERR(00201) Type=0.0 Nibs=2
eINVST =  000ED TI%R6S        -
eINVUS =  0002E TI%R6S        -
eINX   =  00015 TI%R6S        -
eION   =  00043 NZ%ERR        - F0430 NZ%ERR(00026) Type=0.0 Nibs=2
                              + F065A NZ%ERR(00250) Type=0.0 Nibs=2
eIVARG =  0000B TI%R6S        - F0627 NZ%ERR(0021D) Type=0.0 Nibs=2
eIVSAR =  00033 TI%R6S        -
eIVSOP =  00035 TI%R6S        -
eIVSTA =  00034 TI%R6S        -
eIVTAB =  00030 TI%R6S        -
eL2LNG =  00041 TI%R6S        -
eLNO   =  0000C TI%R6S        -
eLOBAT =  00016 TI%R6S        -
eLOG-  =  0000D TI%R6S        -
eLPERR =  00026 NZ%ERR        -
eLTIMO =  00023 NZ%ERR        - F05A7 NZ%ERR(0019D) Type=0.0 Nibs=2
                              + F05AF NZ%ERR(001A5) Type=0.0 Nibs=2
eMEDIA =  00042 NZ%ERR        - F0477 NZ%ERR(0006D) Type=0.0 Nibs=2
                              + F0482 NZ%ERR(00078) Type=0.0 Nibs=2
                              + F0499 NZ%ERR(0008F) Type=0.0 Nibs=2
eMEM   =  00018 TI%R6S        - F0643 NZ%ERR(00239) Type=0.0 Nibs=2
eMMCOR =  00017 TI%R6S        - F05B7 NZ%ERR(001AD) Type=0.0 Nibs=2
eMPI   =  00019 TI%R6S        -
eMSPAR =  00052 TI%R6S        - F0440 NZ%ERR(00036) Type=0.0 Nibs=2
eMSPAr =  00004 NZ%ERR        - F759B NZ%PAR(0009E) Type=0.0 Nibs=1
                              + F7910 NZ%PAR(00413) Type=0.0 Nibs=1
eNEG^X =  00009 TI%R6S        -
```

```
eNEWTA =  00017 NZ%ERR        - F0B40 NZ%GPR(0037E) Type=0.0 Nibs=1
                              + F4338 NZ%CAS(000A5) Type=0.0 Nibs=1
                              + F4B1C NZ%CAS(00889) Type=0.0 Nibs=1
eNFILE =  00016 NZ%ERR        - F4833 NZ%CAS(005A0) Type=0.0 Nibs=1
                              + F57D2 NZ%HND(006B7) Type=0.0 Nibs=1
                              + F5D8B NZ%HND(00C70) Type=0.0 Nibs=1
eNFOUN =  000E8 TI%R6S        -
eNMBOX =  00039 NZ%ERR        - F1DF8 NZ%BAS(00E5E) Type=0.0 Nibs=1
                              + F3CC0 NZ%BUT(000C9) Type=0.0 Nibs=1
eNNUMR =  00036 NZ%ERR        - F1DE7 NZ%BAS(00E4D) Type=0.0 Nibs=1
                              + F1F04 NZ%BAS(00F6A) Type=0.0 Nibs=1
                              + F2EA3 NZ%UTL(0020D) Type=0.0 Nibs=1
                              + F3FE6 NZ%BUT(003EF) Type=0.0 Nibs=1
                              + F4039 NZ%BUT(00442) Type=0.0 Nibs=1
                              + F6DB9 NZ%LOW(00063) Type=0.0 Nibs=1
eNOASN =  00001 NZ%ERR        - F17C2 NZ%BAS(00828) Type=0.0 Nibs=1
eNODAT =  00020 TI%R6S        -
eNOFND =  00020 NZ%ERR        - F0AB9 NZ%GPR(002F7) Type=0.0 Nibs=1
                              + F1D2A NZ%BAS(00D90) Type=0.0 Nibs=1
eNOLIF =  00013 NZ%ERR        - F4930 NZ%CAS(0069D) Type=0.0 Nibs=1
eNORAM =  0003B NZ%ERR        - F35E3 NZ%BIF(0070C) Type=0.0 Nibs=1
                              + F3EEA NZ%BUT(002F3) Type=0.0 Nibs=1
                              + F4633 NZ%CAS(003A0) Type=0.0 Nibs=1
                              + F613F NZ%CAT(002AE) Type=0.0 Nibs=1
                              + F6F49 NZ%FXQ(00130) Type=0.0 Nibs=1
eNORDY =  00022 NZ%ERR        - F0CE4 NZ%GPR(00522) Type=0.0 Nibs=1
                              + F2A5D SC%ENT(00B29) Type=0.0 Nibs=1
eNOTAP =  00014 NZ%ERR        -
eNOTIN =  00043 TI%R6S        -
eNSVAR =  00033 TI%R6S        -
eNUMIN =  00026 TI%R6S        -
eNVSTA =  00033 TI%R6S        -
eNXwoF =  0002B TI%R6S        -
eOFFED =  0003C NZ%ERR        - F3C4E NZ%BUT(00057) Type=0.0 Nibs=1
eOVFL* =  000F5 TI%R6S        -
eOVFLW =  00002 TI%R6S        -
eOVRUN =  00025 NZ%ERR        -
ePALGN =  0005E TI%R6S        -
ePARSE =  00000 NZ%SYM        - F17C4 NZ%BAS(0082A) Type=0.0 Nibs=1
                              + F34D3 NZ%BIF(005FC) Type=0.0 Nibs=1
                              + F34E7 NZ%BIF(00610) Type=0.0 Nibs=1
ePIL   =  00002 NZ%SYM        - F08AF NZ%GPR(000ED) Type=0.0 Nibs=1
                              + F0ABB NZ%GPR(002F9) Type=0.0 Nibs=1
                              + F0ADE NZ%GPR(0031C) Type=0.0 Nibs=1
                              + F0C77 NZ%GPR(004B5) Type=0.0 Nibs=1
                              + F0CDB NZ%GPR(00519) Type=0.0 Nibs=1
                              + F1D21 NZ%BAS(00D87) Type=0.0 Nibs=1
                              + F1D3E NZ%BAS(00DA4) Type=0.0 Nibs=1
                              + F2460 SC%ENT(0052C) Type=0.0 Nibs=1
                              + F290A SC%ENT(009D6) Type=0.0 Nibs=1
                              + F2A5F SC%ENT(00B2B) Type=0.0 Nibs=1
                              + F34F1 NZ%BIF(0061A) Type=0.0 Nibs=1
                              + F3519 NZ%BIF(00642) Type=0.0 Nibs=1 Offset=      1
                              + F35A7 NZ%BIF(006D0) Type=0.0 Nibs=1
                              + F430B NZ%CAS(00078) Type=0.0 Nibs=1
                              + F53DC NZ%HND(002C1) Type=0.0 Nibs=1
                              + F670B NZ%IOR(00039) Type=0.0 Nibs=1
                              + F685A NZ%IOR(00188) Type=0.0 Nibs=1
ePLLC  =  0005A TI%R6S        -
ePLLC# =  00059 TI%R6S        -
```

```
ePRCER =  00054 TI%R6S        -
ePRMIS =  00024 TI%R6S        -
ePRNEX =  0004C TI%R6S        -
ePROTD =  00042 TI%R6S        -
ePRTCT =  000F8 TI%R6S        -
ePULL  =  000F6 TI%R6S        -
eQUOEX =  0004D TI%R6S        -
eROWRN =  00056 TI%R6S        -
eR1WRN =  00057 TI%R6S        -
eRALGN =  0005D TI%R6S        -
eRANGE =  00038 NZ%ERR        - F0E1E NZ%GPR(0065C) Type=0.0 Nibs=1
                              + F13FF NZ%BAS(00465) Type=0.0 Nibs=1
                              + F141C NZ%BAS(00482) Type=0.0 Nibs=1
                              + F14B0 NZ%BAS(00516) Type=0.0 Nibs=1
                              + F16E6 NZ%BAS(0074C) Type=0.0 Nibs=1
                              + F17BD NZ%BAS(00823) Type=0.0 Nibs=1
                              + F1DCF NZ%BAS(00E35) Type=0.0 Nibs=1
                              + F1EE2 NZ%BAS(00F48) Type=0.0 Nibs=1
                              + F1F0A NZ%BAS(00F70) Type=0.0 Nibs=1
                              + F29B0 SC%ENT(00A7C) Type=0.0 Nibs=1
                              + F2E26 NZ%UTL(00190) Type=0.0 Nibs=1
                              + F2EA9 NZ%UTL(00213) Type=0.0 Nibs=1
                              + F3FFE NZ%BUT(00407) Type=0.0 Nibs=1
                              + F403D NZ%BUT(00446) Type=0.0 Nibs=1
                              + F40BF NZ%BUT(004C8) Type=0.0 Nibs=1
                              + F43C7 NZ%CAS(00134) Type=0.0 Nibs=1
                              + F4869 NZ%CAS(005D6) Type=0.0 Nibs=1
                              + F51DF NZ%HND(000C4) Type=0.0 Nibs=1
                              + F60DA NZ%CAT(00249) Type=0.0 Nibs=1
                              + F6E14 NZ%LOW(000BE) Type=0.0 Nibs=1
                              + F6FB5 NZ%FXQ(0019C) Type=0.0 Nibs=1
                              + F7044 NZ%FXQ(0022B) Type=0.0 Nibs=1
                              + F70A4 NZ%FXQ(0028B) Type=0.0 Nibs=1
                              + F7115 NZ%FXQ(002FC) Type=0.0 Nibs=1
                              + F72D4 NZ%FXQ(004BB) Type=0.0 Nibs=1
                              + F7392 NZ%FXQ(00579) Type=0.0 Nibs=1
eRECOR =  0001D TI%R6S        -
eRECRD =  00019 NZ%ERR        -
eRRORX =  F2D37 NZ%UTL        - F2A61 SC%ENT(00B2D) Type=1.0 Nibs=3 Dist=002D6
eRWERR =  00046 TI%R6S        -
eRwoGS =  0002C TI%R6S        -
eSIGOP =  00013 TI%R6S        -
eSPGNF =  00031 TI%R6S        -
eSQR-  =  0000A TI%R6S        -
eSTALL =  00012 NZ%ERR        -
eSTMNF =  0001E TI%R6S        -
eSTROV =  00025 TI%R6S        -
eSUBSC =  0001C TI%R6S        -
eSYNTX =  0004B TI%R6S        - F0458 NZ%ERR(0004E) Type=0.0 Nibs=2
eSYNTx =  00007 NZ%ERR        - F756C NZ%PAR(0006F) Type=0.0 Nibs=1
                              + F75C8 NZ%PAR(000CB) Type=0.0 Nibs=1
eSYSER =  00017 TI%R6S        -
eSYSer =  0002C NZ%ERR        - F53DA NZ%HND(002BF) Type=0.0 Nibs=1
eTAPE  =  00001 NZ%SYM        - F0B37 NZ%GPR(00375) Type=0.0 Nibs=1
                              + F0B57 NZ%GPR(00395) Type=0.0 Nibs=1
                              + F34EC NZ%BIF(00615) Type=0.0 Nibs=1
                              + F432F NZ%CAS(0009C) Type=0.0 Nibs=1
                              + F4835 NZ%CAS(005A2) Type=0.0 Nibs=1
                              + F492C NZ%CAS(00699) Type=0.0 Nibs=1
                              + F4B13 NZ%CAS(00880) Type=0.0 Nibs=1
```

```
                                        + F4BE3 NZ%CAS(00950) Type=0.0 Nibs=1
                                        + F4DA6 NZ%CAS(00B13) Type=0.0 Nibs=1
                                        + F57D4 NZ%HND(006B9) Type=0.0 Nibs=1
                                        + F57DF NZ%HND(006C4) Type=0.0 Nibs=1
                                        + F5D82 NZ%HND(00C67) Type=0.0 Nibs=1
                                        + F5D9C NZ%HND(00C81) Type=0.0 Nibs=1
eTERM  =  00020 NZ%ERR                  -
eTESTF =  0002D NZ%ERR                  -
eTFFLD =  00038 TI%R6S                  -
eTFM   =  000F1 TI%R6S                  -
eTFWRN =  00058 TI%R6S                  -
eTNINF =  00004 TI%R6S                  -
eTOO   =  000EF TI%R6S                  -
eTOOFI =  00028 TI%R6S                  -
eTOOMI =  00027 TI%R6S                  -
eTRKDN =  00061 TI%R6S                  -
eTRKOF =  000E5 TI%R6S                  -
eTSIZE =  0001C NZ%ERR                  - F4926 NZ%CAS(00693) Type=0.0 Nibs=1
                                        + F57DD NZ%HND(006C2) Type=0.0 Nibs=1
eTUFAS =  00047 TI%R6S                  -
eTUSLO =  00048 TI%R6S                  -
eUALGN =  0005F TI%R6S                  -
eUNEXP =  00027 NZ%ERR                  - F0ADC NZ%GPR(0031A) Type=0.0 Nibs=1
                                        + F0CD5 NZ%GPR(00513) Type=0.0 Nibs=1
                                        + F1D3C NZ%BAS(00DA2) Type=0.0 Nibs=1
                                        + F245A SC%ENT(00526) Type=0.0 Nibs=1
                                        + F6709 NZ%IOR(00037) Type=0.0 Nibs=1
eUNFLW =  00001 TI%R6S                  -
eUNKCD =  00045 TI%R6S                  -
eUNORC =  00014 TI%R6S                  -
eVALGN =  0005C TI%R6S                  -
eVARTY =  00032 TI%R6S                  -
eVFYER =  00044 TI%R6S                  -
eWALGN =  0005B TI%R6S                  -
eWRGNM =  00049 TI%R6S                  -
eXCESS =  00003 NZ%ERR                  -
eXFNNF =  00022 TI%R6S                  -
eXPEXC =  F4107 NZ%BUT                  - F178F NZ%BAS(007F5) Type=1.1 Nibs=4 Dist=02978
                                        + F2244 SC%ENT(00310) Type=1.1 Nibs=4 Dist=01EC3
                                        + F2E7E NZ%UTL(001E8) Type=1.1 Nibs=4 Dist=01289
eXWORD =  00023 TI%R6S                  -
eXXXXX =  00028 NZ%ERR                  -
eZRDIV =  00008 TI%R6S                  -
eZRO/O =  00007 TI%R6S                  -
efPROT =  00010 NZ%ERR                  - F4BE1 NZ%CAS(0094E) Type=0.0 Nibs=1
enull  =  00000 TI%R6S                  -
eu/o   =  000EB TI%R6S                  -

fAOS   =  000DF TI%R6S                  -
fASCII =  00001 TI%R6S                  -
fBASIC =  0E214 TI%R6S                  - F56BC NZ%HND(005A1) Type=0.0 Nibs=4
fBIN   =  0E204 TI%R6S                  -
fDATA  =  0E0F0 TI%R6S                  -
fEOF   =  000FF TI%R6S                  -
fEOR   =  000EF TI%R6S                  -
fEOS   =  0006F TI%R6S                  -
fKEY   =  0E20C TI%R6S                  - F58F3 NZ%HND(007D8) Type=0.0 Nibs=5
fLEX   =  0E208 TI%R6S                  - F0018 NZ%TBL(00010) Type=0.0 Nibs=4
                                        + F5BFB NZ%HND(00AE0) Type=0.0 Nibs=4
fLIF1  =  00001 TI%R6S                  -
```

```
fLTDH  =  F1F2D NZ%BAS        - F3FEB NZ%BUT(003F4) Type=1.1 Nibs=4 Dist=020BE
                              + F4020 NZ%BUT(00429) Type=1.1 Nibs=4 Dist=020F3
                              + F6103 NZ%CAT(00272) Type=1.1 Nibs=4 Dist=041D6
fMOS   =  0007F TI%R6S        -
fPROT  =  F4BDD NZ%CAS        - F5CE4 NZ%HND(00BC9) Type=1.1 Nibs=4 Dist=01107
fSDATA =  0E0D0 TI%R6S        -
fSOS   =  000CF TI%R6S        -
fTEXT  =  00001 TI%R6S        -
fTYPF# =  F5CB0 NZ%HND        - F4BCD NZ%CAS(0093A) Type=1.1 Nibs=4 Dist=010E3
                              + F63F6 NZ%CAT(00565) Type=1.1 Nibs=3 Dist=00746
f1AC   =  FFFC7 TI%R6S        -
f1ALRM =  FFFC4 TI%R6S        -
f1BASE =  FFFF0 TI%R6S        -
f1BAT  =  FFFC3 TI%R6S        -
f1BEEP =  FFFFE TI%R6S        -
f1BPLD =  FFFE7 TI%R6S        -
f1CALC =  FFFC0 TI%R6S        -
f1CLOC =  FFFD3 TI%R6S        -
f1CMDS =  FFFD1 TI%R6S        -
f1CTON =  FFFFD TI%R6S        -
f1CTRL =  FFFD0 TI%R6S        -
f1DG0  =  FFFEF TI%R6S        -
f1DG1  =  FFFEE TI%R6S        -
f1DG2  =  FFFED TI%R6S        -
f1DG3  =  FFFEC TI%R6S        -
f1DORM =  FFFD5 TI%R6S        - F319B NZ%BIF(002C4) Type=0.0 Nibs=2
f1DVZ  =  FFFF9 TI%R6S        -
f1EOT  =  FFFE9 TI%R6S        - F2309 SC%ENT(003D5) Type=0.0 Nibs=2
f1EXAC =  FFFD2 TI%R6S        -
f1EXTD =  FFFEA TI%R6S        - F092C NZ%GPR(0016A) Type=0.0 Nibs=2
f1FXEN =  FFFF3 TI%R6S        -
f1INFR =  FFFF5 TI%R6S        -
f1INX  =  FFFFC TI%R6S        -
f1IVL  =  FFFF8 TI%R6S        -
f1LC   =  FFFF1 TI%R6S        -
f1MKOF =  FFFCE TI%R6S        -
f1NEGR =  FFFF4 TI%R6S        -
f1NOFN =  FFFD6 TI%R6S        -
f1NOPR =  FFFE6 TI%R6S        -
f1NZ4  =  FFFE8 TI%R6S        - F090B NZ%GPR(00149) Type=0.0 Nibs=2
f1OVF  =  FFFFA TI%R6S        -
f1PDWN =  FFFEB TI%R6S        - F304D NZ%BIF(00176) Type=0.0 Nibs=2
f1PRGM =  FFFC2 TI%R6S        -
f1PWDN =  FFFCF TI%R6S        -
f1QIET =  FFFFF TI%R6S        -
f1RAD  =  FFFF6 TI%R6S        -
f1RPTD =  FFFC5 TI%R6S        -
f1RTN  =  FFFD4 TI%R6S        -
f1SCEN =  FFFF2 TI%R6S        -
f1SUSP =  FFFC1 TI%R6S        -
f1TNOF =  FFFCD TI%R6S        -
f1UNF  =  FFFFB TI%R6S        -
f1USER =  FFFF7 TI%R6S        -
f1USRX =  FFFC6 TI%R6S        -
f1VIEW =  FFFCC TI%R6S        -

getdev =  F28F9 SC%ENT        - F2CB4 NZ%UTL(0001E) Type=1.1 Nibs=3 Dist=003B8

hCAT   =  F5E91 NZ%CAT        - F06EA NZ%DIR(00035) Type=1.2 Nibs=5 Dist=057A7
hCAT$  =  F60BF NZ%CAT        - F06EF NZ%DIR(0003A) Type=1.2 Nibs=5 Dist=059D0
```

```
hCOPYx =  F54B8 NZ%HND        - F06F4 NZ%DIR(0003F) Type=1.2 Nibs=5 Dist=04DC4
hCPY5s =  F5C15 NZ%HND        - F23A3 SC%ENT(0046F) Type=1.1 Nibs=4 Dist=03872
                              + F2BE7 SC%ENT(00CB3) Type=1.1 Nibs=4 Dist=0302E
                              + F46B7 NZ%CAS(00424) Type=1.1 Nibs=4 Dist=0155E
                              + F47C0 NZ%CAS(0052D) Type=1.1 Nibs=4 Dist=01455
hCREAT =  F51B3 NZ%HND        - F06F9 NZ%DIR(00044) Type=1.2 Nibs=5 Dist=04ABA
hDIDST =  F35FD NZ%BIF        - F06FE NZ%DIR(00049) Type=1.2 Nibs=5 Dist=02EFF
hENTER =  F1F34 SC%ENT        - F0726 NZ%DIR(00071) Type=1.2 Nibs=5 Dist=0180E
hEXCPT =  F2B0A SC%ENT        - F07B8 NZ%DIR(00103) Type=1.2 Nibs=5 Dist=02352
hFINDF =  F5153 NZ%HND        - F073F NZ%DIR(0008A) Type=1.2 Nibs=5 Dist=04A14
hFPROT =  F5E03 NZ%HND        - F0703 NZ%DIR(0004E) Type=1.2 Nibs=5 Dist=05700
hKYDF  =  F2B98 SC%ENT        - F0753 NZ%DIR(0009E) Type=1.2 Nibs=5 Dist=02445
hPRTCL =  F5438 NZ%HND        - F0712 NZ%DIR(0005D) Type=1.2 Nibs=5 Dist=04D26
hPURGE =  F5CBD NZ%HND        - F071C NZ%DIR(00067) Type=1.2 Nibs=5 Dist=055A1
hRDCBF =  F52C4 NZ%HND        - F0744 NZ%DIR(0008F) Type=1.2 Nibs=5 Dist=04B80
hRDNBF =  F532F NZ%HND        - F0749 NZ%DIR(00094) Type=1.2 Nibs=5 Dist=04BE6
hRENAM =  F5D6E NZ%HND        - F0721 NZ%DIR(0006C) Type=1.2 Nibs=5 Dist=0564D
hVER$  =  F511B NZ%HND        - F06CC NZ%DIR(00017) Type=1.2 Nibs=5 Dist=04A4F
hWRCBF =  F5313 NZ%HND        - F074E NZ%DIR(00099) Type=1.2 Nibs=5 Dist=04BC5
hZERPG =  F2ADD SC%ENT        - F07BD NZ%DIR(00108) Type=1.2 Nibs=5 Dist=02320
hs3BYT =  00007 NZ%SYM        -
hsAWKE =  00002 NZ%SYM        -
hsERRO =  00004 NZ%SYM        -
hsLPRQ =  00005 NZ%SYM        - F1E53 NZ%BAS(00EB9) Type=0.0 Nibs=1
hsMANL =  00006 NZ%SYM        -
hsMGAV =  00000 NZ%SYM        -
hsNRD  =  00001 NZ%SYM        -
hsRQSR =  00003 NZ%SYM        - F2BBF SC%ENT(00C8B) Type=0.0 Nibs=1
                              + F3156 NZ%BIF(0027F) Type=0.0 Nibs=1

i/OFND =  F410E NZ%BUT        - F0946 NZ%GPR(00184) Type=1.1 Nibs=4 Dist=037C8
                              + F17DA NZ%BAS(00840) Type=1.1 Nibs=4 Dist=02934
                              + F4BFE NZ%CAS(0096B) Type=1.1 Nibs=4 Dist=00AF0
                              + F53D1 NZ%HND(002B6) Type=1.1 Nibs=4 Dist=012C3
                              + F5D32 NZ%HND(00C17) Type=1.1 Nibs=4 Dist=01C24

k#-CHR =  00068 TI%R6S        -
k#-LIN =  0006B TI%R6S        -
k#1    =  00027 TI%R6S        -
k#2    =  00028 TI%R6S        -
k#3    =  00029 TI%R6S        -
k#ATTN =  0002B TI%R6S        -
k#BKSP =  00067 TI%R6S        -
k#BOT  =  000A3 TI%R6S        - F5FD6 NZ%CAT(00145) Type=0.0 Nibs=2
k#CALC =  0006F TI%R6S        -
k#CONT =  00070 TI%R6S        -
k#CTRL =  0009E TI%R6S        -
k#DOWN =  00033 TI%R6S        - F5FCC NZ%CAT(0013B) Type=0.0 Nibs=2
k#EOL  =  00026 TI%R6S        -
k#FLFT =  0009F TI%R6S        -
k#FRT  =  000A0 TI%R6S        -
k#GON  =  0009B TI%R6S        -
k#I/R  =  00069 TI%R6S        -
k#LAST =  000A4 TI%R6S        -
k#LC   =  0006A TI%R6S        -
k#LERR =  000A1 TI%R6S        -
k#LFT  =  0002F TI%R6S        -
k#OFF  =  00063 TI%R6S        -
k#RT   =  00030 TI%R6S        -
k#RUN  =  0002E TI%R6S        -
```

```
k#SST  =  00066 TI%R6S        -
k#TOP  =  000A2 TI%R6S        - F5FDB NZ%CAT(0014A) Type=0.0 Nibs=2
k#UP   =  00032 TI%R6S        - F5FD1 NZ%CAT(00140) Type=0.0 Nibs=2
k#USER =  0006D TI%R6S        -
k#USEX =  000A5 TI%R6S        -
k#VIEW =  0006E TI%R6S        -
kc-CHR =  00000 TI%R6S        -
kc-LIN =  00004 TI%R6S        -
kcATTN =  0000E TI%R6S        -
kcBKSP =  00007 TI%R6S        -
kcBOT  =  00015 TI%R6S        -
kcCALC =  00017 TI%R6S        -
kcCONT =  00010 TI%R6S        -
kcCTRL =  0000A TI%R6S        -
kcDOWN =  00013 TI%R6S        -
kcEOL  =  0000D TI%R6S        -
kcFLFT =  00005 TI%R6S        -
kcFRT  =  00006 TI%R6S        -
kcGON  =  00016 TI%R6S        -
kcI/R  =  00002 TI%R6S        -
kcLAST =  00019 TI%R6S        -
kcLC   =  00001 TI%R6S        -
kcLERR =  0001A TI%R6S        -
kcLFT  =  00008 TI%R6S        -
kcOFF  =  00018 TI%R6S        -
kcRT   =  00009 TI%R6S        -
kcRUN  =  0000F TI%R6S        -
kcSST  =  00011 TI%R6S        -
kcTOP  =  00014 TI%R6S        -
kcUP   =  00012 TI%R6S        -
kcUSER =  00003 TI%R6S        -
kcUSEX =  0000C TI%R6S        -
kcVIEW =  0000B TI%R6S        -

lACCSb =  00001 TI%R6S        -
lAp    =  00010 TI%R6S        -
lBPOSp =  00005 TI%R6S        -
lCOPYb =  00001 TI%R6S        -
lCPOSb =  00006 TI%R6S        -
lD0p   =  00005 TI%R6S        -
lD1p   =  00005 TI%R6S        -
lDATEh =  00006 TI%R6S        -
lDBEGb =  0000B TI%R6S        -
lDEVC  =  00005 TI%R6S        - F426D NZ%BUT(00676) Type=0.0 Nibs=1 Offset=      3
                              + F4278 NZ%BUT(00681) Type=0.0 Nibs=1 Offset=      3
lDEVCb =  00001 TI%R6S        -
lDLENb =  00006 TI%R6S        -
lDp    =  00010 TI%R6S        -
lEOL   =  00002 TI%R6S        -
lFBEGb =  00006 TI%R6S        -
lFBF#b =  00003 TI%R6S        -
lFIB   =  0003F TI%R6S        -
lFIL#b =  00002 TI%R6S        -
lFILSV =  00032 TI%R6S        -
lFLAGh =  00002 TI%R6S        -
lFLENh =  00005 TI%R6S        - F5665 NZ%HND(0054A) Type=0.0 Nibs=1
                              + F56A1 NZ%HND(00586) Type=0.0 Nibs=1
                              + F574B NZ%HND(00630) Type=0.0 Nibs=1 Offset=     -1
                              + F5758 NZ%HND(0063D) Type=0.0 Nibs=1 Offset=     -1
                              + F575B NZ%HND(00640) Type=0.0 Nibs=1 Offset=      8
```

```
                                + F58B4 NZ%HND(00799) Type=0.0 Nibs=2
                                + F59E3 NZ%HND(008C8) Type=0.0 Nibs=1
                                + F64AA NZ%CAT(00619) Type=0.0 Nibs=2
 lFNAM+ =  00004 TI%R6S         -
 lFNAM8 =  00010 TI%R6S         -
 lFNAMh =  00010 TI%R6S         -
 lFSIZb =  00006 TI%R6S         -
 lFTYPb =  00004 TI%R6S         -
 lFTYPh =  00004 TI%R6S         -
 lLXADR =  00005 TI%R6S         -
 lLXENT =  0000B TI%R6S         -
 lLXFAD =  00005 TI%R6S         -
 lLXID  =  00002 TI%R6S         -
 lLXTKR =  00004 TI%R6S         -
 lMSGp  =  00004 TI%R6S         -
 lPOL#p =  00005 TI%R6S         -
 lPOLLp =  00005 TI%R6S         -
 lPOLSV =  0003E TI%R6S         - F4264 NZ%BUT(0066D) Type=0.0 Nibs=2
 lPOLra =  00006 TI%R6S         -
 lPROTb =  00001 TI%R6S         -
 lREC#b =  00004 TI%R6S         -
 lRECLb =  00004 TI%R6S         -
 lRLENb =  00005 TI%R6S         -
 lRTN1p =  00005 TI%R6S         -
 lRTN2p =  00005 TI%R6S         -
 lRTN3p =  00005 TI%R6S         -
 lSHLNb =  00002 TI%R6S         -
 lSPDTB =  0004E TI%R6S         -
 lSPDn  =  00001 TI%R6S         -
 lSPDn2 =  00001 TI%R6S         -
 lTEXTp =  00004 TI%R6S         -
 lTIMEh =  00004 TI%R6S         -

 mADDRL =  05000 NZ%SYM         - F0CF3 NZ%GPR(00531) Type=0.0 Nibs=4
 mADDRM =  02000 NZ%SYM         - F0D28 NZ%GPR(00566) Type=0.0 Nibs=4 Offset=      4
                                + F0D39 NZ%GPR(00577) Type=0.0 Nibs=4 Offset=      2
 mADDRT =  04000 NZ%SYM         - F0D46 NZ%GPR(00584) Type=0.0 Nibs=4
 mAUTO  =  00009 NZ%SYM         -
 mAUTOA =  00070 NZ%SYM         - F0934 NZ%GPR(00172) Type=0.0 Nibs=2 Offset=      1
                                + F0952 NZ%GPR(00190) Type=0.0 Nibs=2
 mAUTOE =  00007 NZ%SYM         -
 mAUTOS =  00071 NZ%SYM         -
 mCLRBF =  000F8 NZ%SYM         -
 mCLRCA =  F0000 NZ%SYM         - F2AAA SC%ENT(00B76) Type=0.0 Nibs=6
 mCMD2  =  00014 NZ%SYM         -
 mCMD3  =  00140 NZ%SYM         - F6BC2 NZ%IOR(004F0) Type=0.0 Nibs=3 Offset=     10
                                + F6BD1 NZ%IOR(004FF) Type=0.0 Nibs=3 Offset=     12
 mCMDf  =  01400 NZ%SYM         - F1648 NZ%BAS(006AE) Type=0.0 Nibs=4
 mCSRQ  =  00004 NZ%SYM         -
 mDATA2 =  00010 NZ%SYM         -
 mDATAf =  01000 NZ%SYM         -
 mEAR   =  01418 NZ%SYM         -
 mENDM  =  00003 NZ%SYM         - F4E5F NZ%CAS(00BCC) Type=0.0 Nibs=2
 mENDf  =  01200 NZ%SYM         - F4EE1 NZ%CAS(00C4E) Type=0.0 Nibs=4
                                + F50DB NZ%CAS(00E48) Type=0.0 Nibs=4
 mERSTS =  00006 NZ%SYM         - F682A NZ%IOR(00158) Type=0.0 Nibs=2
 mETE   =  01541 NZ%SYM         -
 mETO   =  01540 NZ%SYM         -
 mFIND1 =  00006 NZ%SYM         - F09CD NZ%GPR(0020B) Type=0.0 Nibs=1
 mFINDP =  06000 NZ%SYM         - F0B11 NZ%GPR(0034F) Type=0.0 Nibs=4 Offset=     16
```

```
mFRAME =  01000 NZXSYM        -
mGETCA =  0000C NZXSYM        - F0A9A NZXGPR(002D8) Type=0.0 Nibs=2
mIDYf  =  01600 NZXSYM        -
mIFC   =  01490 NZXSYM        -
mINCCA =  0000D NZXSYM        - F0AC2 NZXGPR(00300) Type=0.0 Nibs=2
mMADDR =  0000E NZXSYM        -
mMANUL =  00008 NZXSYM        -
mNOP   =  00000 NZXSYM        -
mPDLOP =  00030 NZXSYM        - F307B NZXBIF(001A4) Type=0.0 Nibs=2
mPULOP =  000FE NZXSYM        - F08BD NZXGPR(000FB) Type=0.0 Nibs=2
mRDADR =  00001 NZXSYM        -
mRDYf  =  01500 NZXSYM        -
mREADC =  000FC NZXSYM        - F1DA2 NZXBAS(00E08) Type=0.0 Nibs=2
mREADI =  000FB NZXSYM        - F1D4F NZXBAS(00DB5) Type=0.0 Nibs=2
mRSTCA =  0000B NZXSYM        - F0A37 NZXGPR(00275) Type=0.0 Nibs=2
mRdMem =  00000 NZXSYM        -
mSAI   =  00000 NZXSYM        - F0C9F NZXGPR(004DD) Type=0.0 Nibs=6 Offset=      2
mSCOPE =  00801 NZXSYM        -
mSDA   =  00000 NZXSYM        - F0B72 NZXGPR(003B0) Type=0.0 Nibs=6 Offset=      8
                              + F4363 NZXCAS(000D0) Type=0.0 Nibs=6 Offset=      2
                              + F449F NZXCAS(0020C) Type=0.0 Nibs=6 Offset=     12
                              + F45BA NZXCAS(00327) Type=0.0 Nibs=6 Offset=    256
                              + F4769 NZXCAS(004D6) Type=0.0 Nibs=6 Offset=     32
                              + F48BE NZXCAS(0062B) Type=0.0 Nibs=6 Offset=     32
                              + F48EC NZXCAS(00659) Type=0.0 Nibs=6 Offset=     24
                              + F496A NZXCAS(006D7) Type=0.0 Nibs=6 Offset=     12
                              + F4B3E NZXCAS(008AB) Type=0.0 Nibs=6 Offset=     32
mSDA@5 =  00008 NZXSYM        - F5C19 NZXHND(00AFE) Type=0.0 Nibs=1
mSDI   =  00000 NZXSYM        - F68AD NZXIOR(001DB) Type=0.0 Nibs=6 Offset=      8
mSETAI =  30321 NZXSYM        - F3217 NZXBIF(00340) Type=0.0 Nibs=6
mSETAl =  30120 NZXSYM        - F3208 NZXBIF(00331) Type=0.0 Nibs=6
mSETCA =  00F01 NZXSYM        - F2AC9 SCXENT(00B95) Type=0.0 Nibs=4
mSETDI =  30011 NZXSYM        - F324B NZXBIF(00374) Type=0.0 Nibs=6
mSETDR =  30000 NZXSYM        -
mSETDl =  30610 NZXSYM        - F3226 NZXBIF(0034F) Type=0.0 Nibs=6
mSETFC =  00000 NZXSYM        - F2CBC NZXUTL(00026) Type=0.0 Nibs=6 Offset=1048575
mSETIC =  0F600 NZXSYM        - F1762 NZXBAS(007C8) Type=0.0 Nibs=4
mSETIM =  0FA00 NZXSYM        - F29E5 SCXENT(00AB1) Type=0.0 Nibs=4
                              + F2AF2 SCXENT(00BBE) Type=0.0 Nibs=4
                              + F2B7A SCXENT(00C46) Type=0.0 Nibs=4
mSETIT =  0F700 NZXSYM        - F31F9 NZXBIF(00322) Type=0.0 Nibs=4 Offset=     50
mSETST =  30041 NZXSYM        - F2935 SCXENT(00A01) Type=0.0 Nibs=2
mSETSl =  30140 NZXSYM        - F2923 SCXENT(009EF) Type=0.0 Nibs=6
mSETTC =  0F500 NZXSYM        - F251B SCXENT(005E7) Type=0.0 Nibs=4
mSETTM =  0F400 NZXSYM        - F237C SCXENT(00448) Type=0.0 Nibs=4 Offset=     12
                              + F24D2 SCXENT(0059E) Type=0.0 Nibs=4
                              + F24DC SCXENT(005A8) Type=0.0 Nibs=4 Offset=      8
                              + F250E SCXENT(005DA) Type=0.0 Nibs=4 Offset=      1
mSETTO =  00000 NZXSYM        -
mSFC@5 =  0000E NZXSYM        - F24F6 SCXENT(005C2) Type=0.0 Nibs=1
                              + F5C25 NZXHND(00B0A) Type=0.0 Nibs=1
mSPDIS =  0FF00 NZXSYM        -
mSPEN  =  0FF01 NZXSYM        -
mSPTO  =  0F900 NZXSYM        -
mSSRQ  =  00005 NZXSYM        -
mSST   =  00000 NZXSYM        - F1BB8 NZXBAS(00C1E) Type=0.0 Nibs=6 Offset=      8
                              + F429E NZXCAS(0000B) Type=0.0 Nibs=6 Offset=      1
mSTATS =  00002 NZXSYM        - F6820 NZXIOR(0014E) Type=0.0 Nibs=2
mSTO@5 =  0000D NZXSYM        - F1774 NZXBAS(007DA) Type=0.0 Nibs=1
mSTS@4 =  000F3 NZXSYM        - F293B SCXENT(00A07) Type=0.0 Nibs=2
```

```
mSTSTC =  00201 NZ%SYM        - F3166 NZ%BIF(0028F) Type=0.0 Nibs=4
mTAKEC =  00F03 NZ%SYM        -
mTAKEI =  F0390 NZ%SYM        - F08D1 NZ%GPR(0010F) Type=0.0 Nibs=6
mTAKEO =  F0310 NZ%SYM        -
mTCT   =  00000 NZ%SYM        - F2A41 SC%ENT(00B0D) Type=0.0 Nibs=6
mTEST  =  000F2 NZ%SYM        -
mUNADM =  02010 NZ%SYM        -
mUNL   =  0143F NZ%SYM        - F0D04 NZ%GPR(00542) Type=0.0 Nibs=4
mUNT   =  0145F NZ%SYM        - F24EA SC%ENT(005B6) Type=0.0 Nibs=4
mUPDSC =  00A00 NZ%SYM        -
mWrMem =  10000 NZ%SYM        -
maddrL =  00002 NZ%SYM        -
maddrT =  00004 NZ%SYM        -

nXTSTM =  F1780 NZ%BAS        - F0851 NZ%GPR(0008F) Type=1.0 Nibs=4 Dist=00F2F
                              + F1F8A SC%ENT(00056) Type=1.0 Nibs=4 Dist=0080A
                              + F6E0F NZ%LOW(000B9) Type=1.0 Nibs=4 Dist=0568F

o41sod =  00005 TI%R6S        - F573C NZ%HND(00621) Type=0.0 Nibs=1
oACCSb =  0000B TI%R6S        - F5308 NZ%HND(001ED) Type=0.0 Nibs=1 Offset=     -1
                              + F53BD NZ%HND(002A2) Type=0.0 Nibs=1 Offset=     -1
oAp    =  0003E TI%R6S        -
oBNsod =  00011 TI%R6S        -
oBPOSp =  00005 TI%R6S        -
oBSsod =  00011 TI%R6S        -
oCOPYb =  0000A TI%R6S        -
oCPOSb =  00028 TI%R6S        -
oD0p   =  00019 TI%R6S        -
oD1p   =  0001E TI%R6S        -
oDATEh =  0001A TI%R6S        -
oDAsod =  0000D TI%R6S        - F59DC NZ%HND(008C1) Type=0.0 Nibs=1 Offset=     -5
                              + F5A6B NZ%HND(00950) Type=0.0 Nibs=1 Offset=     -5
oDBEGb =  00015 TI%R6S        -
oDEVCb =  0000C TI%R6S        - F539A NZ%HND(0027F) Type=0.0 Nibs=1 Offset=     -1
oDLENb =  0002E TI%R6S        -
oDp    =  0002E TI%R6S        -
oFBEGb =  0000D TI%R6S        - F5D45 NZ%HND(00C2A) Type=0.0 Nibs=1 Offset=     -1
oFBF#b =  00002 TI%R6S        -
oFIL#b =  00000 TI%R6S        -
oFLAGh =  00014 TI%R6S        - F5640 NZ%HND(00525) Type=0.0 Nibs=1 Offset=     -1
                              + F59AE NZ%HND(00893) Type=0.0 Nibs=2
oFLENh =  00020 TI%R6S        - F56EE NZ%HND(005D3) Type=0.0 Nibs=2 Offset=     17
oFLSTr =  00031 TI%R6S        -
oFNAMh =  00000 TI%R6S        -
oFSIZb =  00039 TI%R6S        -
oFT-FL =  00010 TI%R6S        -
oFTYPb =  00005 TI%R6S        -
oFTYPh =  00010 TI%R6S        - F563D NZ%HND(00522) Type=0.0 Nibs=1 Offset=     -1
                              + F5BEC NZ%HND(00AD1) Type=0.0 Nibs=2
oIMPLh =  00025 TI%R6S        - F5A35 NZ%HND(0091A) Type=0.0 Nibs=2
oINHS  =  00008 NZ%IOR        - F31D3 NZ%BIF(002FC) Type=0.0 Nibs=1 Offset=     -1
                              + F31D9 NZ%BIF(00302) Type=0.0 Nibs=1 Offset=     -1
oINST  =  00009 NZ%IOR        -
oKYsod =  00005 TI%R6S        -
oLXsod =  00005 TI%R6S        -
oMAINT =  0005D TI%R6S        -
oMSGPT =  00009 TI%R6S        -
oOUTHS =  00007 NZ%IOR        - F31E0 NZ%BIF(00309) Type=0.0 Nibs=1 Offset=     -1
                              + F31E7 NZ%BIF(00310) Type=0.0 Nibs=1 Offset=     -1
oOUTST =  00006 NZ%IOR        -
```

```
oPOL#p  =  0000A TI%R6S        -
oPROTb  =  00009 TI%R6S        -
oREC#b  =  00020 TI%R6S        -
oRECLb  =  00024 TI%R6S        -
oRLENb  =  00034 TI%R6S        -
oRTN1p  =  0000A TI%R6S        -
oRTN2p  =  0000F TI%R6S        -
oRTN3p  =  00014 TI%R6S        -
oSHLNb  =  00013 TI%R6S        -
oSPDTB  =  00111 TI%R6S        -
oSPDn2  =  0000E TI%R6S        -
oSUBLn  =  00025 TI%R6S        -
oTIMEh  =  00016 TI%R6S        -
oTXsod  =  00005 TI%R6S        -
oUT1TK  =  F7ABE NZ%PAR        - F7D68 NZ%DEC(00195) Type=1.1 Nibs=3 Dist=002AA
                               + F7E32 NZ%DEC(0025F) Type=1.1 Nibs=3 Dist=00374
oUT2TC  =  F7AC5 NZ%PAR        - F7CFB NZ%DEC(00128) Type=1.0 Nibs=3 Dist=00236
oUT3TK  =  F7ACF NZ%PAR        -
oUTNBS  =  F7AD9 NZ%PAR        -

p3DATA  =  0000F NZ%SYM        - F07CD NZ%GPR(0000B) Type=0.0 Nibs=1
pACK    =  00000 NZ%SYM        - F0815 NZ%GPR(00053) Type=0.0 Nibs=1
                               + F2A56 SC%ENT(00B22) Type=0.0 Nibs=1
pADDR   =  00004 NZ%SYM        - F07EF NZ%GPR(0002D) Type=0.0 Nibs=1
                               + F096A NZ%GPR(001A8) Type=0.0 Nibs=1
                               + F09D7 NZ%GPR(00215) Type=0.0 Nibs=1
                               + F0AA5 NZ%GPR(002E3) Type=0.0 Nibs=1
                               + F0ACD NZ%GPR(0030B) Type=0.0 Nibs=1
                               + F0B29 NZ%GPR(00367) Type=0.0 Nibs=1
pBSCen  =  000F5 TI%R6S        -
pBSCex  =  000F6 TI%R6S        -
pCALRS  =  00036 TI%R6S        -
pCALSV  =  00037 TI%R6S        -
pCAT    =  00006 TI%R6S        -
pCAT$   =  00007 TI%R6S        -
pCLDST  =  000FF TI%R6S        -
pCMD    =  0000C NZ%SYM        -
pCMPLX  =  00038 TI%R6S        -
pCONFG  =  000FB TI%R6S        -
pCOPYx  =  00008 TI%R6S        -
pCRDAB  =  00033 TI%R6S        -
pCREAT  =  00009 TI%R6S        -
pCRT=8  =  00023 TI%R6S        -
pCURSR  =  00029 TI%R6S        -
pDATA   =  0000B NZ%SYM        - F07E6 NZ%GPR(00024) Type=0.0 Nibs=1
                               + F0CB4 NZ%GPR(004F2) Type=0.0 Nibs=1
                               + F686C NZ%IOR(0019A) Type=0.0 Nibs=1
pDATLN  =  0002A TI%R6S        -
pDEVCp  =  00001 TI%R6S        -
pDIAGL  =  00003 NZ%SYM        - F1D5F NZ%BAS(00DC5) Type=0.0 Nibs=1
pDIAGR  =  00002 NZ%SYM        -
pDIDST  =  0000A TI%R6S        -
pDSWKY  =  000FD TI%R6S        -
pDSWNK  =  000FE TI%R6S        -
pEDIT   =  0002B TI%R6S        -
pENTER  =  00012 TI%R6S        -
pEOFIL  =  00025 TI%R6S        -
pEOT    =  00006 NZ%SYM        - F0827 NZ%GPR(00065) Type=0.0 Nibs=1
                               + F0CCC NZ%GPR(0050A) Type=0.0 Nibs=1
                               + F2417 SC%ENT(004E3) Type=0.0 Nibs=1
```

```
                                        + F44EF NZ%CAS(0025C) Type=0.0 Nibs=1
                                        + F5BB9 NZ%HND(00A9E) Type=0.0 Nibs=1
                                        + F6723 NZ%IOR(00051) Type=0.0 Nibs=1
                                        + F6889 NZ%IOR(001B7) Type=0.0 Nibs=1
                                        + F695B NZ%IOR(00289) Type=0.0 Nibs=1
pERROR =  000F2 TI%R6S                  -
pETE   =  00009 NZ%SYM                  - F081E NZ%GPR(0005C) Type=0.0 Nibs=1
pExcpt =  000F8 TI%R6S                  -
pFASCH =  0002C TI%R6S                  -
pFILDC =  00002 TI%R6S                  -
pFILXQ =  00003 TI%R6S                  -
pFINDF =  00017 TI%R6S                  -
pFNIN  =  0003D TI%R6S                  -
pFNOUT =  0003E TI%R6S                  -
pFPROT =  0000B TI%R6S                  -
pFSPCp =  00004 TI%R6S                  -
pFSPCx =  00005 TI%R6S                  -
pFTYPE =  0002D TI%R6S                  -
pHALTD =  00007 NZ%SYM                  - F0830 NZ%GPR(0006E) Type=0.0 Nibs=1
pIDY   =  0000E NZ%SYM                  -
pIFC   =  00005 NZ%SYM                  - F0839 NZ%GPR(00077) Type=0.0 Nibs=1
pIMCHR =  0001E TI%R6S                  -
pIMXCH =  0001F TI%R6S                  -
pIMXQT =  0001D TI%R6S                  -
pIMbck =  00020 TI%R6S                  -
pIMcpı =  00021 TI%R6S                  -
pIMcpш =  00022 TI%R6S                  -
pKYDF  =  0001B TI%R6S                  -
pLIST  =  0000C TI%R6S                  -
pLIST2 =  0002E TI%R6S                  -
pMEM   =  000F1 TI%R6S                  -
pMERGE =  0000D TI%R6S                  -
pMNLP  =  000FA TI%R6S                  -
pMRGE2 =  0002F TI%R6S                  -
pPARSE =  000F4 TI%R6S                  -
pPRGPR =  00032 TI%R6S                  -
pPRIN# =  00026 TI%R6S                  -
pPRTCL =  0000E TI%R6S                  -
pPRTIS =  0000F TI%R6S                  -
pPURGE =  00010 TI%R6S                  -
pPWROF =  000FC TI%R6S                  -
pRCRD  =  00034 TI%R6S                  -
pRDCBF =  00018 TI%R6S                  -
pRDNBF =  00019 TI%R6S                  -
pRDY   =  0000D NZ%SYM                  -
pREAD# =  00027 TI%R6S                  -
pREN   =  00039 TI%R6S                  -
pRNAME =  00011 TI%R6S                  -
pRTNIp =  0003A TI%R6S                  -
pRUNft =  00030 TI%R6S                  -
pRUNnB =  00031 TI%R6S                  -
pSREC# =  00028 TI%R6S                  -
pSREQ  =  000F9 TI%R6S                  -
pSTATE =  00001 NZ%SYM                  - F08C8 NZ%GPR(00106) Type=0.0 Nibs=1
                                        + F0AD5 NZ%GPR(00313) Type=0.0 Nibs=1
                                        + F0CD1 NZ%GPR(0050F) Type=0.0 Nibs=1
                                        + F244F SC%ENT(0051B) Type=0.0 Nibs=1
                                        + F671B NZ%IOR(00049) Type=0.0 Nibs=1
                                        + F6842 NZ%IOR(00170) Type=0.0 Nibs=1
                                        + F6956 NZ%IOR(00284) Type=0.0 Nibs=1
```

```
pTERM  =  00008 NZ%SYM        - F0842 NZ%GPR(00080) Type=0.0 Nibs=1
                              + F2425 SC%ENT(004F1) Type=0.0 Nibs=1
                              + F5BB4 NZ%HND(00A99) Type=0.0 Nibs=1
                              + F6728 NZ%IOR(00056) Type=0.0 Nibs=1
pTEST  =  000F0 TI%R6S        -
pTIMR#  =  0003B TI%R6S       -
pTRANS =  000EF TI%R6S        -
pTRFMx =  0003C TI%R6S        -
pUTYPE =  0000A NZ%SYM        - F0848 NZ%GPR(00086) Type=0.0 Nibs=1
pVER$  =  00000 TI%R6S        -
pWARN  =  000F3 TI%R6S        -
pWCRD  =  00035 TI%R6S        -
pWCRD8 =  00024 TI%R6S        -
pWRCBF =  0001A TI%R6S        -
pWTKY  =  0001C TI%R6S        -
pZERPG =  000F7 TI%R6S        -

rEV$   =  F61B2 NZ%CAT        - F200B SC%ENT(000D7) Type=1.1 Nibs=4 Dist=041A7
                              + F20C7 SC%ENT(00193) Type=1.1 Nibs=4 Dist=040EB

s3BYTE =  00003 NZ%SYM        -
sARITH =  00007 TI%R6S        -
sBYEx  =  00000 TI%R6S        -
sC/P   =  00001 TI%R6S        -
sCARD  =  00002 TI%R6S        - F54C5 NZ%HND(003AA) Type=0.0 Nibs=1
sCARDC =  00008 TI%R6S        -
sCHAIN =  0000B TI%R6S        -
sCONT  =  0000A TI%R6S        -
sCONTK =  00009 TI%R6S        -
sCONTR =  00000 NZ%SYM        - F0C4C NZ%GPR(0048A) Type=0.0 Nibs=1
                              + F3074 NZ%BIF(0019D) Type=0.0 Nibs=1
sCURBT =  00003 TI%R6S        -
sCURUD =  00004 TI%R6S        -
sCURUP =  00002 TI%R6S        -
sCntg  =  00002 TI%R6S        -
sCplxP =  00007 TI%R6S        -
sDATAO =  00009 NZ%SYM        -
sDATAV =  00008 NZ%SYM        - F2BCE SC%ENT(00C9A) Type=0.0 Nibs=1
                              + F3191 NZ%BIF(002BA) Type=0.0 Nibs=1
sDEST  =  00003 TI%R6S        - F4273 NZ%BUT(0067C) Type=0.0 Nibs=1
                              + F5A04 NZ%HND(008E9) Type=0.0 Nibs=1
                              + F5C58 NZ%HND(00B3D) Type=0.0 Nibs=1
                              + F5DA7 NZ%HND(00C8C) Type=0.0 Nibs=1
                              + F5DEB NZ%HND(00CD0) Type=0.0 Nibs=1
sDIAsr =  00001 NZ%SYM        - F2F77 NZ%BIF(000A0) Type=0.0 Nibs=1
                              + F3128 NZ%BIF(00251) Type=0.0 Nibs=1
sDevOK =  00008 NZ%SYM        - F35E8 NZ%BIF(00711) Type=0.0 Nibs=1
                              + F6E8B NZ%FXQ(00072) Type=0.0 Nibs=1
                              + F6EC2 NZ%FXQ(000A9) Type=0.0 Nibs=1
sENDx  =  00001 TI%R6S        -
sEOF   =  00007 TI%R6S        -
sERROR =  00000 NZ%SYM        - F6792 NZ%IOR(000C0) Type=0.0 Nibs=1
                              + F6AFC NZ%IOR(0042A) Type=0.0 Nibs=1
sEXTDV =  00000 TI%R6S        - F54BA NZ%HND(0039F) Type=0.0 Nibs=1
sEXTGS =  00005 TI%R6S        - F2B90 SC%ENT(00C5C) Type=0.0 Nibs=1
sFLAG? =  F0989 NZ%GPR        - F230D SC%ENT(003D9) Type=1.1 Nibs=4 Dist=01984
                              + F3051 NZ%BIF(0017A) Type=1.1 Nibs=4 Dist=026C8
                              + F319F NZ%BIF(002C8) Type=1.1 Nibs=4 Dist=02816
sFOUND =  0000A TI%R6S        -
sFirst =  00000 NZ%SYM        - F742D NZ%FXQ(00614) Type=0.0 Nibs=1
```

```
                                + F744D NZ%FXQ(00634) Type=0.0 Nibs=1
                                + F7459 NZ%FXQ(00640) Type=0.0 Nibs=1
                                + F747D NZ%FXQ(00664) Type=0.0 Nibs=1
sGOSUB =  00003 TI%R6S          -
sI/OBF =  0000A TI%R6S          -
sINFRD =  0000A TI%R6S          -
sINTR  =  00004 NZ%SYM          - F2B1B SC%ENT(00BE7) Type=0.0 Nibs=1
                                + F3186 NZ%BIF(002AF) Type=0.0 Nibs=1
sINX   =  00005 TI%R6S          -
sIRAM  =  00002 TI%R6S          -
sIX    =  00007 TI%R6S          -
sInit  =  00003 TI%R6S          -
sKEYS  =  00005 TI%R6S          -
sLISTR =  00001 NZ%SYM          -
sLOCKD =  0000B NZ%SYM          -
sLoop? =  00005 NZ%SYM          - F4736 NZ%CAS(004A3) Type=0.0 Nibs=1
                                + F473D NZ%CAS(004AA) Type=0.0 Nibs=1
                                + F475D NZ%CAS(004CA) Type=0.0 Nibs=1
                                + F47B5 NZ%CAS(00522) Type=0.0 Nibs=1
                                + F47C9 NZ%CAS(00536) Type=0.0 Nibs=1
                                + F5A12 NZ%HND(008F7) Type=0.0 Nibs=1
                                + F5AB8 NZ%HND(0099D) Type=0.0 Nibs=1
                                + F5B77 NZ%HND(00A5C) Type=0.0 Nibs=1
sMAINc =  00005 TI%R6S          -
sMANUL =  00002 NZ%SYM          - F0C2C NZ%GPR(0046A) Type=0.0 Nibs=1
sMULT  =  00008 TI%R6S          -
sNAPRS =  F52F5 NZ%HND          - F360B NZ%BIF(00734) Type=1.1 Nibs=4 Dist=01CEA
sNEGRD =  0000B TI%R6S          -
sNoChn =  00002 TI%R6S          -
sONERR =  00004 TI%R6S          -
sONTMR =  00006 TI%R6S          -
sOVERW =  00008 NZ%SYM          - F4BB8 NZ%CAS(00925) Type=0.0 Nibs=1
                                + F52B8 NZ%HND(0019D) Type=0.0 Nibs=1
                                + F55BA NZ%HND(0049F) Type=0.0 Nibs=1
                                + F5775 NZ%HND(0065A) Type=0.0 Nibs=1
sPCRD  =  00008 TI%R6S          -
sPOLLE =  00006 NZ%SYM          -
sPRGCF =  0000B TI%R6S          -
sPRIVT =  0000B NZ%SYM          - F5E2A NZ%HND(00D0F) Type=0.0 Nibs=1
sRAD   =  00009 TI%R6S          -
sRDX   =  0000B TI%R6S          -
sREADI =  00004 TI%R6S          -
sRENAM =  00006 TI%R6S          -
sRENUM =  00008 TI%R6S          -
sRESTR =  0000A TI%R6S          -
sRETRN =  00000 TI%R6S          -
sRFILE =  00008 TI%R6S          -
sRMOTE =  0000A NZ%SYM          - F3196 NZ%BIF(002BF) Type=0.0 Nibs=1
sRUNBn =  00004 TI%R6S          -
sRUNDC =  00007 TI%R6S          -
sReadd =  00004 NZ%SYM          - F0885 NZ%GPR(000C3) Type=0.0 Nibs=1
                                + F08B8 NZ%GPR(000F6) Type=0.0 Nibs=1
                                + F08FA NZ%GPR(00138) Type=0.0 Nibs=1
                                + F0929 NZ%GPR(00167) Type=0.0 Nibs=1
                                + F19A9 NZ%BAS(00A0F) Type=0.0 Nibs=1
                                + F1A6B NZ%BAS(00AD1) Type=0.0 Nibs=1
sSCNTR =  00003 NZ%SYM          -
sSIGN  =  00009 TI%R6S          -
sSRQIN =  00001 NZ%SYM          -
sSST   =  00002 TI%R6S          -
```

```
sSSTdc =  00001 TI%R6S        -
sSTAND =  00007 NZ%SYM        -
sSTAT  =  00006 TI%R6S        -
sSTK   =  00007 NZ%SYM        - F19D5 NZ%BAS(00A3B) Type=0.0 Nibs=1
                              + F1C8B NZ%BAS(00CF1) Type=0.0 Nibs=1
                              + F352E NZ%BIF(00657) Type=0.0 Nibs=1
                              + F3F1B NZ%BUT(00324) Type=0.0 Nibs=1
                              + F3F4A NZ%BUT(00353) Type=0.0 Nibs=1
                              + F3F64 NZ%BUT(0036D) Type=0.0 Nibs=1
                              + F3F94 NZ%BUT(0039D) Type=0.0 Nibs=1
                              + F3FC4 NZ%BUT(003CD) Type=0.0 Nibs=1
                              + F4007 NZ%BUT(00410) Type=0.0 Nibs=1
                              + F4044 NZ%BUT(0044D) Type=0.0 Nibs=1
                              + F6E24 NZ%FXQ(0000B) Type=0.0 Nibs=1
                              + F6EC5 NZ%FXQ(000AC) Type=0.0 Nibs=1
                              + F73EC NZ%FXQ(005D3) Type=0.0 Nibs=1
sSTOP  =  00005 TI%R6S        -
sSpec1 =  00006 TI%R6S        -
sTALKA =  00002 NZ%SYM        - F50D1 NZ%CAS(00E3E) Type=0.0 Nibs=1
sUNCNF =  00005 NZ%SYM        - F0901 NZ%GPR(0013F) Type=0.0 Nibs=1
sUNDEF =  00001 TI%R6S        - F5508 NZ%HND(003ED) Type=0.0 Nibs=1
sUNSEC =  0000A NZ%SYM        - F5E3B NZ%HND(00D20) Type=0.0 Nibs=1
sXCPT  =  00004 TI%R6S        -
sXQT   =  00000 TI%R6S        -
sXWORD =  00009 TI%R6S        -

t!     =  000FC TI%R6S        -
t%     =  00085 TI%R6S        - F7318 NZ%FXQ(004FF) Type=0.0 Nibs=2
                              + F78DD NZ%PAR(003E0) Type=0.0 Nibs=2
                              + F7DE3 NZ%DEC(00210) Type=0.0 Nibs=2
t&     =  00089 TI%R6S        -
t*     =  00083 TI%R6S        - F7336 NZ%FXQ(0051D) Type=0.0 Nibs=2
                              + F76AD NZ%PAR(001B0) Type=0.0 Nibs=2
                              + F78BF NZ%PAR(003C2) Type=0.0 Nibs=2
                              + F7DB8 NZ%DEC(001E5) Type=0.0 Nibs=2
t+     =  00087 TI%R6S        -
t-     =  00082 TI%R6S        -
t/     =  00084 TI%R6S        -
t@     =  000F4 TI%R6S        - F7561 NZ%PAR(00064) Type=0.0 Nibs=2
tABS   =  000A2 TI%R6S        -
tACOS  =  0009A TI%R6S        -
tADD   =  000D5 TI%R6S        -
tADIG0 =  00060 TI%R6S        -
tADIG1 =  00061 TI%R6S        -
tADIG2 =  00062 TI%R6S        -
tADIG3 =  00063 TI%R6S        -
tADIG4 =  00064 TI%R6S        -
tADIG5 =  00065 TI%R6S        -
tADIG6 =  00066 TI%R6S        -
tADIG7 =  00067 TI%R6S        -
tADIG8 =  00068 TI%R6S        -
tADIG9 =  00069 TI%R6S        -
tALL   =  000F8 TI%R6S        -
tAND   =  0008B TI%R6S        -
tANGLE =  601B3 TI%R6S        -
tARRAY =  0007D TI%R6S        -
tASIN  =  00099 TI%R6S        -
tATAN  =  0009B TI%R6S        -
tAUTO  =  000EE TI%R6S        -
tBASE  =  000E9 TI%R6S        -
```

```
tBEEP  =  000E8 TI%R6S        -
tBIG   =  00010 TI%R6S        -
tCALL  =  000F9 TI%R6S        -
tCARD  =  000D0 TI%R6S        -
tCAT   =  000EC TI%R6S        -
tCEIL  =  00072 TI%R6S        -
tCFLAG =  000FA TI%R6S        -
tCHR$  =  000A4 TI%R6S        -
tCLOCK =  501EF TI%R6S        -
tCMPLX =  0007A TI%R6S        -
tCNTRL =  00023 NZ%TBL        - F7B7B NZ%PAR(0067E) Type=0.0 Nibs=2
tCOLON =  000E2 TI%R6S        - F2D56 NZ%UTL(000C0) Type=0.0 Nibs=2
                              + F3F23 NZ%BUT(0032C) Type=0.0 Nibs=2
                              + F726E NZ%FXQ(00455) Type=0.0 Nibs=2
                              + F72AA NZ%FXQ(00491) Type=0.0 Nibs=2
                              + F7892 NZ%PAR(00395) Type=0.0 Nibs=2
                              + F7BC6 NZ%PAR(006C9) Type=0.0 Nibs=2
                              + F7EE8 NZ%DEC(00315) Type=0.0 Nibs=2
tCOMMA =  000F1 TI%R6S        - F146B NZ%BAS(004D1) Type=0.0 Nibs=2
                              + F1598 NZ%BAS(005FE) Type=0.0 Nibs=2
                              + F1710 NZ%BAS(00776) Type=0.0 Nibs=2
                              + F2A03 SC%ENT(00ACF) Type=0.0 Nibs=2
                              + F2CF6 NZ%UTL(00060) Type=0.0 Nibs=2
                              + F2D4D NZ%UTL(000B7) Type=0.0 Nibs=2
                              + F6DD2 NZ%LOW(0007C) Type=0.0 Nibs=2
                              + F7614 NZ%PAR(00117) Type=0.0 Nibs=2
                              + F7832 NZ%PAR(00335) Type=0.0 Nibs=2
                              + F79D8 NZ%PAR(004DB) Type=0.0 Nibs=2
                              + F7D85 NZ%DEC(001B2) Type=0.0 Nibs=2
                              + F7EDD NZ%DEC(0030A) Type=0.0 Nibs=2
tCOPY  =  000B5 TI%R6S        -
tCOS   =  00097 TI%R6S        -
tCVAL  =  000E1 TI%R6S        -
tDATA  =  000C6 TI%R6S        -
tDATE  =  00077 TI%R6S        -
tDATE$ =  00078 TI%R6S        -
tDEF   =  000B9 TI%R6S        -
tDEG   =  0006F TI%R6S        -
tDEGRE =  000D3 TI%R6S        -
tDELAY =  000D6 TI%R6S        -
tDELET =  000B7 TI%R6S        -
tDIM   =  000CC TI%R6S        -
tDISP  =  000C5 TI%R6S        -
tDIV   =  00086 TI%R6S        -
tDMYAR =  0007E TI%R6S        -
tDSTRY =  000BE TI%R6S        -
tDVZ   =  000B1 TI%R6S        -
tEDIT  =  000B8 TI%R6S        -
tELSE  =  000F5 TI%R6S        -
tEND   =  000DA TI%R6S        -
tENDDF =  000BA TI%R6S        -
tENDSB =  000C2 TI%R6S        -
tENTER =  4FFEF TI%R6S        - F2570 SC%ENT(0063C) Type=0.0 Nibs=6
tEOL   =  000F0 TI%R6S        - F2B6C SC%ENT(00C38) Type=0.0 Nibs=2
tEPS   =  00071 TI%R6S        -
tERRL  =  00075 TI%R6S        -
tERRN  =  00076 TI%R6S        -
tERROR =  000E3 TI%R6S        -
tEXOR  =  0008C TI%R6S        -
tEXP   =  00094 TI%R6S        -
```

```
tEXTIF  =  000F4 TI%R6S         -
tEXTND  =  601EF TI%R6S         -
tFACT   =  000A8 TI%R6S         -
tFETCH  =  000C8 TI%R6S         -
tFFN    =  000B4 TI%R6S         -
tFLOW   =  901EF TI%R6S         -
tFLT1   =  0001D TI%R6S         -
tFLT10  =  00014 TI%R6S         -
tFLT11  =  00013 TI%R6S         -
tFLT12  =  00012 TI%R6S         -
tFLT2   =  0001C TI%R6S         -
tFLT3   =  0001B TI%R6S         -
tFLT4   =  0001A TI%R6S         -
tFLT5   =  00019 TI%R6S         -
tFLT6   =  00018 TI%R6S         -
tFLT7   =  00017 TI%R6S         -
tFLT8   =  00016 TI%R6S         -
tFLT9   =  00015 TI%R6S         -
tFN     =  0007C TI%R6S         -
tFOR    =  000C3 TI%R6S         -
tFP     =  0006B TI%R6S         -
tGOSUB  =  000DC TI%R6S         -
tGOTO   =  000DD TI%R6S         -
tIF     =  000DF TI%R6S         -
tIMAGE  =  000FF TI%R6S         -
tIN     =  000F2 TI%R6S         -
tINF    =  00070 TI%R6S         -
tINPUT  =  000C9 TI%R6S         -
tINT    =  0009C TI%R6S         -
tINT10  =  00004 TI%R6S         -
tINT11  =  00003 TI%R6S         -
tINT12  =  00002 TI%R6S         -
tINT2   =  0000C TI%R6S         -
tINT3   =  0000B TI%R6S         -
tINT4   =  0000A TI%R6S         -
tINT5   =  00009 TI%R6S         -
tINT6   =  00008 TI%R6S         -
tINT7   =  00007 TI%R6S         -
tINT8   =  00006 TI%R6S         -
tINT9   =  00005 TI%R6S         -
tINTEG  =  000CA TI%R6S         -
tINTO   =  E01EF TI%R6S         -
tINTR   =  015FF TI%R6S         -
tINTRR  =  00026 NZ%TBL         - F7650 NZ%PAR(00153) Type=0.0 Nibs=2
                                + F7680 NZ%PAR(00183) Type=0.0 Nibs=2
                                + F7B4F NZ%PAR(00652) Type=0.0 Nibs=2
tINX    =  000B2 TI%R6S         -
tIO     =  00024 NZ%TBL         - F7663 NZ%PAR(00166) Type=0.0 Nibs=2
tIP     =  0006A TI%R6S         -
tIS     =  000E7 TI%R6S         - F7503 NZ%PAR(00006) Type=0.0 Nibs=2
tISUB$  =  000A7 TI%R6S         -
tIVL    =  000AE TI%R6S         -
tKEY    =  000E5 TI%R6S         -
tKEY$   =  00073 TI%R6S         -
tKEYS   =  000CF TI%R6S         -
tLBLRF  =  0000E TI%R6S         -
tLBLST  =  000F6 TI%R6S         -
tLEN    =  000A9 TI%R6S         -
tLET    =  000C0 TI%R6S         -
tLINE#  =  0000F TI%R6S         -
```

```
tLINPT =  000BF TI%R6S        -
tLIST  =  000BB TI%R6S        -
tLITRL =  000C4 TI%R6S        - F355E NZ%BIF(00687) Type=0.0 Nibs=2
                              + F3F2F NZ%BUT(00338) Type=0.0 Nibs=2
                              + F7281 NZ%FXQ(00468) Type=0.0 Nibs=2
                              + F7866 NZ%PAR(00369) Type=0.0 Nibs=2
                              + F797F NZ%PAR(00482) Type=0.0 Nibs=2
                              + F7E19 NZ%DEC(00246) Type=0.0 Nibs=2
tLN    =  00091 TI%R6S        -
tLOCK0 =  00025 NZ%TBL        - F1524 NZ%BAS(0058A) Type=0.0 Nibs=2
                              + F75F6 NZ%PAR(000F9) Type=0.0 Nibs=2
                              + F7C9B NZ%DEC(000C8) Type=0.0 Nibs=2
tLOG   =  00090 TI%R6S        -
tLOG10 =  00093 TI%R6S        -
tLPRP  =  000AA TI%R6S        -
tLR    =  000B6 TI%R6S        -
tMAIN  =  000D2 TI%R6S        -
tMATH  =  601EF TI%R6S        -
tMAX   =  000AD TI%R6S        -
tMAXRL =  0006C TI%R6S        -
tMEAN  =  0009D TI%R6S        -
tMIN   =  000AC TI%R6S        -
tMOD   =  00074 TI%R6S        -
tNAME  =  000BD TI%R6S        -
tNEAR  =  C01EF TI%R6S        -
tNEG   =  D01EF TI%R6S        -
tNEXT  =  000C4 TI%R6S        -
tNOT   =  00081 TI%R6S        -
tNUM   =  000A3 TI%R6S        -
tOFF   =  000E1 TI%R6S        - F16BD NZ%BAS(00723) Type=0.0 Nibs=2
                              + F75DA NZ%PAR(000DD) Type=0.0 Nibs=2
                              + F7BA3 NZ%PAR(006A6) Type=0.0 Nibs=2
                              + F7C83 NZ%DEC(000B0) Type=0.0 Nibs=2
tON    =  000E0 TI%R6S        - F16C6 NZ%BAS(0072C) Type=0.0 Nibs=2
                              + F2A8F SC%ENT(00B5B) Type=0.0 Nibs=2
                              + F75D5 NZ%PAR(000D8) Type=0.0 Nibs=2
                              + F7B9E NZ%PAR(006A1) Type=0.0 Nibs=2
                              + F7C7A NZ%DEC(000A7) Type=0.0 Nibs=2
tOPT'N =  000ED TI%R6S        -
tOR    =  0008D TI%R6S        -
tOVF   =  000AF TI%R6S        -
tPAUSE =  000D7 TI%R6S        -
tPCRD  =  E01EF TI%R6S        -
tPI    =  00079 TI%R6S        -
tPORT  =  000D1 TI%R6S        -
tPOS   =  201B3 TI%R6S        -
tPREDV =  0009F TI%R6S        -
tPRINT =  000CD TI%R6S        -
tPRMEN =  000F8 TI%R6S        -
tPRMST =  000F3 TI%R6S        -
tPURGE =  000EB TI%R6S        -
tRAD   =  0006E TI%R6S        -
tRDIAN =  000D4 TI%R6S        -
tREAD  =  000C7 TI%R6S        -
tREAL  =  000BC TI%R6S        -
tRELOP =  0008A TI%R6S        -
tREM   =  000E6 TI%R6S        -
tRES   =  0007F TI%R6S        -
tRESTR =  000DE TI%R6S        -
tRETRN =  000DB TI%R6S        -
```

```
tRFILE =  000DE TI%R6S        -
tRMD   =  0006D TI%R6S        -
tRND   =  000A0 TI%R6S        -
tROUND =  C01EF TI%R6S        -
tRUN   =  000FE TI%R6S        -
tSDEV  =  0009E TI%R6S        -
tSEMIC =  000F2 TI%R6S        - F2DF6 NZ%UTL(00160) Type=0.0 Nibs=2
                              + F72FF NZ%FXQ(004E6) Type=0.0 Nibs=2
                              + F735D NZ%FXQ(00544) Type=0.0 Nibs=2
                              + F7534 NZ%PAR(00037) Type=0.0 Nibs=2
                              + F7556 NZ%PAR(00059) Type=0.0 Nibs=2
                              + F7744 NZ%PAR(00247) Type=0.0 Nibs=2
                              + F7753 NZ%PAR(00256) Type=0.0 Nibs=2
                              + F7894 NZ%PAR(00397) Type=0.0 Nibs=2
                              + F7927 NZ%PAR(0042A) Type=0.0 Nibs=2
                              + F7D41 NZ%DEC(0016E) Type=0.0 Nibs=2
                              + F7E44 NZ%DEC(00271) Type=0.0 Nibs=2
                              + F7E5D NZ%DEC(0028A) Type=0.0 Nibs=2
tSFLAG =  000FB TI%R6S        -
tSGN   =  000A1 TI%R6S        -
tSHORT =  000CB TI%R6S        -
tSIN   =  00096 TI%R6S        -
tSMALL =  00011 TI%R6S        -
tSQR   =  00092 TI%R6S        -
tSTAT  =  000CE TI%R6S        -
tSTEP  =  000F6 TI%R6S        -
tSTOP  =  000D9 TI%R6S        -
tSTR$  =  000A6 TI%R6S        -
tSUB   =  000C1 TI%R6S        -
tSVAR  =  0002D TI%R6S        -
tTAB   =  000F7 TI%R6S        -
tTAN   =  00098 TI%R6S        -
tTHEN  =  000F4 TI%R6S        -
tTIME  =  0007B TI%R6S        -
tTIME$ =  00095 TI%R6S        -
tTIMER =  000E4 TI%R6S        -
tTO    =  000F3 TI%R6S        -
tTRACE =  000EA TI%R6S        -
tUNF   =  000B0 TI%R6S        -
tUPRC$ =  000AB TI%R6S        -
tUSER  =  000E2 TI%R6S        -
tUSING =  000FD TI%R6S        - F1FA9 SC%ENT(00075) Type=0.0 Nibs=2
                              + F754D NZ%PAR(00050) Type=0.0 Nibs=2
tVAL   =  000A5 TI%R6S        -
tVARS  =  B01EF TI%R6S        -
tWAIT  =  000D8 TI%R6S        -
tXFN   =  000B3 TI%R6S        -
tXWORD =  000EF TI%R6S        - F1520 NZ%BAS(00586) Type=0.0 Nibs=2
                              + F1930 NZ%BAS(00996) Type=0.0 Nibs=2
                              + F75F2 NZ%PAR(000F5) Type=0.0 Nibs=2
                              + F764C NZ%PAR(0014F) Type=0.0 Nibs=2
                              + F765F NZ%PAR(00162) Type=0.0 Nibs=2
                              + F767C NZ%PAR(0017F) Type=0.0 Nibs=2
                              + F7B4B NZ%PAR(0064E) Type=0.0 Nibs=2
                              + F7B77 NZ%PAR(0067A) Type=0.0 Nibs=2
                              + F7C97 NZ%DEC(000C4) Type=0.0 Nibs=2
                              + F7CDB NZ%DEC(00108) Type=0.0 Nibs=2
tZ     =  0005A TI%R6S        -
tZERO  =  C01EF TI%R6S        -
t^     =  00080 TI%R6S        -
```

```
uALit  =  000F7 TI%R6S        -
uCPLXC =  000EE TI%R6S        - F2627 SC%ENT(006F3) Type=0.0 Nibs=2
uDELIM =  000F4 TI%R6S        - F260E SC%ENT(006DA) Type=0.0 Nibs=2
uHKB^  =  000F6 TI%R6S        - F259E SC%ENT(0066A) Type=0.0 Nibs=2
                              + F266B SC%ENT(00737) Type=0.0 Nibs=2
uIMXCH =  000D4 TI%R6S        -
uIMbck =  000DC TI%R6S        -
uIMend =  000F0 TI%R6S        - F2609 SC%ENT(006D5) Type=0.0 Nibs=2
uIMsta =  000DE TI%R6S        -
uJMPdl =  000DB TI%R6S        -
uJMPst =  000DA TI%R6S        -
uJMP{} =  000D9 TI%R6S        -
uLOOPB =  000D2 TI%R6S        - F25E1 SC%ENT(006AD) Type=0.0 Nibs=2
uLOOPP =  000EF TI%R6S        - F25FF SC%ENT(006CB) Type=0.0 Nibs=2
uLOOPS =  000D3 TI%R6S        - F25FA SC%ENT(006C6) Type=0.0 Nibs=2
uMODES =  0BDB1 TI%R6S        -
uMULT  =  000D1 TI%R6S        - F25DC SC%ENT(006A8) Type=0.0 Nibs=2
uNUMEn =  000FC TI%R6S        -
uNUMEs =  000FD TI%R6S        -
uNUMFn =  000FA TI%R6S        -
uNUMFs =  000FB TI%R6S        -
uNUMNn =  000F8 TI%R6S        -
uNUMNs =  000F9 TI%R6S        -
uOPNM- =  000DF TI%R6S        -
uOPNNM =  000D8 TI%R6S        -
uOPNWM =  000E0 TI%R6S        -
uRES12 =  0C994 TI%R6S        -
uRESD1 =  0E1EE TI%R6S        -
uRESNX =  0C9BD TI%R6S        -
uRESTP =  000F1 TI%R6S        - F2604 SC%ENT(006D0) Type=0.0 Nibs=2
uRESXT =  0C9C1 TI%R6S        -
uRND>P =  0C9CF TI%R6S        -
uSTRPT =  000D0 TI%R6S        - F25D7 SC%ENT(006A3) Type=0.0 Nibs=2
uTEST  =  0D435 TI%R6S        -

vDEVID =  75048 NZ%SYM        - F3235 NZ%BIF(0035E) Type=0.0 Nibs=8

xANGLE =  00006 TI%R6S        -
xCLOCK =  00015 TI%R6S        -
xEXTND =  00026 TI%R6S        -
xFLOW  =  00029 TI%R6S        -
xINTO  =  0002E TI%R6S        -
xMATH  =  00036 TI%R6S        -
xNEAR  =  0003C TI%R6S        -
xNEG   =  0003D TI%R6S        -
xPCRD  =  0003E TI%R6S        -
xPOS   =  00042 TI%R6S        -
xROUND =  0004C TI%R6S        -
xVARS  =  0005B TI%R6S        -
xZERO  =  0001C TI%R6S        -
xromFF *  F0095 NZ%TBL(0008D) -
```

Saturn Hex Code Listing

```
F0000 - 00000000 840594C4 25F4D402 802E0000 21421048 6DF70FF1 06200000 0000F004
F0040 - 502702B0 C12C12C1 2EC0C12C 121F0C12 C12A1193 1C121713 D1B02C12 C12C12C1
F0080 - 2C12C120 4610E730 52600F00 ECD10FE1 061E10FD 20CED10F C303DD10 FB4031E1
F00C0 - 0F27061B 10F290B6 A10F3805 6B10FED1 DBA10F28 1D5C10F1 717AC10F CF14FC10

F0100 - FEC02531 0D450874 10D0007B 810DA11C 0810D2C1 75810D1F 02A610DA 212BF00D
F0140 - 1C051E10 D3213782 0C3D1B4B 20D5B1C6 C60D061D FF00D1A0 2DF00DA4 1DC110D9
F0180 - 3135010D 4A1E8720 DCF03831 0D5913D3 10DB025B 310D551F 4820D2B0 72820DBE
F01C0 - 1EE410D1 609A820D AE000000 09010000 00FD0000 000B1435 359474E4 F0B2494E

F0200 - 414E4441 0B2494E4 34D40520 B2494E45 4F42530B 2494E494 F4254052 49445509
F0240 - 34C45414 25E0D34F 4E44525F 4C432D44 54651444 442560B4 45465149 44480B44
F0280 - 54659444 4270D449 43505C41 49591B54 E41424C4 5412954E 44554254 1D94E494
F02C0 - 459414C4 D0794E44 52562394 F4427C49 43545219 C4F43414 C4D1DC4F 434B4F45

F0300 - 545525F4 6464013F 4E451BF4 55450555 4531D051 434B4449 425B1705 1434B4A1
F0340 - 70514353 502D0525 94E44554 2581D255 41444444 434B0F25 54144494 E44525A0
F0380 - B2554D4F 44554E1D 25541555 543545C1 92554355 44571D25 543545F4 25541173
F03C0 - 554E4446 193505F4 C4C490D3 54514E44 4249522B 35451445 5535C0D4 52594747

F0400 - 45425F11 FF103401 00484059 4C402C51 10514353 59474E4D 34C8030E E4C8040E
F0440 - 25C8050E 15C8060E 05C8070E B4C8001E D3C71116 54E64602 F46602D2 4CB021EC
F0480 - ED24C803 1D21CF04 12E4F602 D24C8061 E93C8071 D21C8081 D21C8091 D21C80A1
F04C0 - D21C91C1 73596A75 602F6660 2EAEC80E 1EB3C32F 1BD44962 7563647F 62797026

F0500 - 457C6C6C 8002E04C B122D148 E4F64702 25561646 97CC132A C4F6F607 022427F6
F0540 - B656E6C3 142D0445 42727F62 7C8052D4 2C8062D4 2CF172A5 5E656870 75636475
F0580 - 64602D04 C8082D42 C1192ECE 3D4F6465 6C80A2D3 2C80B2D3 2C80C2E7 1C72D2BF
F05C0 - 3556C666 D2475637 47026616 96C65646 C11F2D14 34597075 6C414361 426F6274

F0600 - 75646C41 53ECED14 33507563 6C8063EF 1C8083EB 0C41936E 4F602C4F 6F607C80
F0640 - B3E81C71 C3625543 545F4255 4D34C610 47D45637 37167656 02C41146 44566796
F0680 - 365602C2 1245D456 469657D6 CA134902 94F402E4 56564656 46CFF20D 231E1048
F06C0 - B5606EB0 7690F4A4 01E1702D 6704B000 2817034E 207A7500 D9504CD4 0ABA40FF

F0700 - E2000750 78000280 0062D40A 88001A55 0D4650E0 81046000 F5000A50 00550004
F0740 - 1A4008B4 06EB405C B4054420 7300040E 1007C9C5 C5C9D513 706147C1 07135D90
F0780 - 603BED31 908B5400 07DCF247 20EC720A 7820E882 08E72093 92066920 25320023
F07C0 - 20A4680F F5602F03 0B86B418 6A200B80 D25752B0 386A8024 0B030B80 D2880D32

F0800 - 190E340B 873A30B8 0D089042 88360290 38826026 03884602 70388160 25038858
F0840 - 028030B2 A0278208 CF2F0108 75001180 172108EB 7C14007E 84821018 CE7338EF
F0880 - 83384476 834007F5 354296BB 131F7963 2131F996 39030922 026AB087 46131EF7
F08C0 - BB34C189 17164C03 50930F08 E8726400 75005A00 28CB3F58 E72F5400 874620B8

F0900 - 65200B46 6318E717 05F08E37 72795F5E 48E66728 5431AE70 50311752 28ECBC53
F0940 - 20188E4C 738EBAC5 3107540E 606781F0 77913400 88452AC3 20310E0E FFB66461
F0980 - 03DF134D B8DC4631 6741DB31 F7967F02 F303AC7D 250331F9 967A02F3 0459E31F
F09C0 - 196734AE 92330670 B2400884 B20B84A0 BB47DFF2 80D480F2 AE2A36A3 60E3FD72

F0A00 - 00368A02 0979900D 909BFAFF 80FEAFF2 031F3963 6066B07E C240031B 08EE6164
F0A40 - 0073AE40 0C6C6A66 4B5F3DB8 0D380C5A B225A978 E63E5400 94BB4970 E0AFB80D
F0A80 - E914A325 A0F523AC 3F7B4720 31C07ED1 40088423 613F8E43 A1400203 00220220
F0AC0 - 31D076B1 40088450 5698916D 20307220 231F5967 0FADB3B0 20202020 20280DEA

F0B00 - 99AF524A 83AB3203 30106F7F 7AABF3F3 7E514008 849A8E16 73571881 0080F088
F0B40 - 72080F04 B5D08E57 73590891 B402228E 56064008 ED27347E 35800008 8EC1D540
```

```
F0B80 - 03308002 3916717D 832B914C 0AC3B476 3FE71714 00278E80 0640024B 074606F4
F0BC0 - F6AEE86C 62061361 B244F215 64134079 4AC0B464 60240203 061361BC A7F20B15

F0C00 - E00B1340 78720003 8EA2035D 0028E450 34008E2E B5872648 E9A52400 7DAC5B08
F0C40 - E7DB5400 0B870D20 B061BCA7 F215620B 85A0B154 27A0C070 30B30922 020B03F2
F0C80 - F2747040 07580400 613B7890 400D0352 0000B8EA AE577DF4 0088B518 AC20F0F0
F0CC0 - AEA57E20 038969F8 91C02780 C0220280 D48821F5 ED721040 0330005A BB8C5BE5

F0D00 - 2033F341 61FF8C53 E58C2DA5 74EF4007 ECF40033 40026DCF 7CCF4003 3200279B
F0D40 - F4003300 046CAFAF ABF4BF47 1BF400D6 96EEE010 42034017 20D12490 8A0C1A0C
F0D80 - 55F0D136 883B01A8 E3059188 2B01A460 05B08816 019A0136 55CD9C2A C603D1F2
F0DC0 - F22005A2 E470E557 F3261023 A0E400A3 156FAF1A F2203010 5A0C480A 7156FA80

F0E00 - 97C60040 10C411A7 6A76A76A 7656D284 5EAF1DCA F08AA008 0FE20570 0CBF1B8A
F0E40 - 57FA7043 1B0656F8 AE6E80DE 02AF0A7C 80DE0379 10500331 6A77C104 003102B6
F0E80 - A033314A 56900330 3939E200 F6BB6B62 0114F171 BF0BF0AE A0D880CE 01D68148
F0EC0 - 14764E40 00D880CE 01810810 D67E2E40 00D880CE 017E1ED2 4000D880 2F018108

F0F00 - 10810810 81081081 08100181 48148148 14814814 81401812 81281281 28128128
F0F40 - 12812018 16816816 81681681 68160120 3F020202 02020202 02011F99 5F2011F4
F0F80 - 95F2017A EF147137 0177EF63 FFAC0661 08E51421 F497F2AC 0A4C15F6 DA8EC2D2
F0FC0 - 5F396C03 B2454377 00500000 30713717 41338CFF 321BBB8F 21460A7A EF210D00

F1000 - 1BBB8F20 B15C20B8 46859B44 450849D7 94A50856 71584DB9 6B61879A 07DAC670
F1040 - 0761644A 866C0ABB 15D25201 BBB8F214 60A7D7F7 00007DA3 441000CA 03D30000
F1080 - 91367116 068E76B2 DB135848 8E11954F 07AF5071 360A0189 00F61557 CC514F80
F10C0 - D01D3814 F96A4188 290AF277 858E187F 0342B607 24608E72 D54141F3 88F2AF01

F1100 - 59A7A557 4951FA59 F215F61E C78F15D6 1CB31F21 4D1618D7 3F7160D4 B9A600C3
F1140 - 6034D87F 251215D0 8EB3E169 E497A60E 93603449 7F21368E 95121368 EC9C58E6
F1180 - 7128E274 55518AFA 588555AF 2A7EAC27 EB413334 D87F28A6 6A8EBD02 D21458E7
F11C0 - C0230765 8F041A60 0E7608EE 3C57E004 90630200 06B14400 8EBF0340 023304A8

F1200 - 78E1D634 00AF9108 AFB1098E 75634009 0D31850A FB7F44F6 8AAA7840 17315F3E
F1240 - 6F28AA76 DD8E3683 DD521AFF 7E14CE71 14AFFF6C E8AAF271 90400AF9 7AF3D58E
F1280 - F28375F3 AFD5F0F6 72404006 E7FD4814 8E8E5340 0657F860 21AF97DB 3E9F28AA
F12C0 - F08EA473 7730400A F9766CD0 ABA8E6EF 24002473 107CF928 79737D53 400248CA

F1300 - C8574534 00237D53 400AF97C 53F2C67A E9400218 CD1D38E6 EA576BE6 21003A86
F1340 - 0F666066 EF493118 AF58EA15 34A21737 F90E6F28 AE606970 7F704F0A F471ABD5
F1380 - 7F604F48 A0E473E2 D610B1CB D671AB8E 76D31188 16AD28E6 5C34027B 92491119
F13C0 - 10A8ED32 34A07830 628F6032 7A308E3D 43667F8E 281347E6 093D2790 08AA6028

F1400 - 028C0FC3 110701B1 1B746223 A1A2846B 74EAD68E 7863DA10 0AF85001 197C32CE
F1440 - 7F221094 E903EE76 0021608E 44A54F47 E1278322 0311F14A 96251110 2CA800C5
F1480 - AF100426 161DB135 8ED4E111 874E1AF4 8E7AE18E 52B24C52 88A88F8E 05E1137D
F14C0 - 78EBCE11 08715AAF 88E7AE11 202BA9CA FC120AF8 8EF83F4A 18EAFD24 118E62E2

F1500 - 4806C312 F6BF0187 607D060A FA15A535 FEFF5297 6D171613 41111014 51658E93
F1540 - 416F5034 103906E3 0677608C 06034808 00692016 7603B060 342929F6 410C4760
F1580 - E9060344 04107001 14514A31 1F96622A C2AC78E5 66F49581 3F3F3C7C 7DB6C008
F15C0 - E45854F3 8E133175 B015D61D 3915F292 A218E717 F4B17650 44170908 217C9983

F1600 - 1A0248CA BE178701 4706AF27 64007DA7 D60147C6 4718E6C5 296C5017 1790047C
F1640 - 8C902F33 004114F8 C06558CC 3C28CBB6 F8C8E728 C9F828CB 45581262 C881662D
F1680 - 88CEDC11 F178F201 1F198F20 18CB0D18 C13D1FC5 6032F508 EA371AC7 14A311E9
F16C0 - 6211310E 96602D36 480340D7 00D731E1 D5417286 D1FACB7F 8F779040 F8E39C1D

F1700 - 6D7D1E57 67F14A31 1F966F31 6117F7C6 045CD6D7 8E46C1AC 7AF0DAAF 2DB8FB7C
F1740 - E097A50B 74D8AE09 7C3920AC B738643C 33006FAE 972EE43B 2530DDB8 E8D3543A
```

```
F1780 - 8D84A808 C8F558E4 79270FF4 00323000 5A3AD6A3 6044017B 775908A8 40032802
F17C0 - 301206F3 E01560D8 E5032018 8E039251 EAF21371 3413510B D1E51471 7396E5FC

F1800 - DD98E3B5 F04718E8 E947F15D 521AC38E F2E41712 03F44566 79636568 23715571
F1840 - 7F3F9202 16373796 76E61557 17F39564 6D0A0FF1 5D91D198 FE0C103F 44566796
F1880 - 36560232 1FB98F21 55717F33 020215D3 1138EC56 F05E404D 88E566F1 30163132
F18C0 - 10321AC3 8E58D420 35D372A3 15D51751 4696A831 45171F6F 696E9031 0214D171

F1900 - 3772D0A0 FF15D71D B98FE0C1 06C5F626 E8B36022 D5014A31 FE966111 FD86F2D2
F1940 - 1455D21F CA7F2157 20B85B0B 15528E82 9115720B 8480B155 26E0E983 60432608
F1980 - E6FF01FC A7F2D215 D08E2F81 15720B85 80B15528 54D38E4D EE4C08E6 D51648C6
F19C0 - 54C34360 4DC508E6 45286781 7DBC8EAD 91DB1087 C7C5D076 31AC26F8 F31A2962

F1A00 - EE8E9781 AF2A7E15 DD17D15D DD231A7D 5320188F D7911490 8DA93901 37135134
F1A40 - AF22815C E16E0C56 F14C1BCA 7F2D2154 2AC2BF3E 78548E41 EE5606E9 0BF71691
F1A80 - 4E189D5C D721C8E5 691118D7 7BCB5606 0AB8EDB3 F5B131A3 966F57EA B4858E4A
F1AC0 - 3F4F48EA 54F789B4 118EC306 4A08E1E4 2D08EF09 18EF14F1 5931738E CF81776B

F1B00 - 4E931C29 6680A6D5 882580C1 067A0007 80D160EA 20320188 D14A11C1 17841485
F1B40 - AC38ABB0 8E14D447 4ACB80DF 17F89031 1C1149BF 4BF40D5C EA4680DF AF280F25
F1B80 - 50B56A0E 8E7CCE7D 0B6CF16E 6AC1174E 04D393F6 065C08ED 71F4B235 8000098E
F1BC0 - 6DC44A19 4BFDAF22 7A966680 C1176A04 55D22031 F10EF78E 8C1FD420 320E30EF

F1C00 - 7BB681E8 EFA1F042 0AF2D6F2 F2AE98ED 23F94A70 BF6E697E 135C36FC 9C117540
F1C40 - 4048AB92 8E640F42 38A8B1AF 2D68E5F0 58EEEBE7 43A6F027 3005DEAF 22E31190
F1C80 - 36389153 7857B04A 64968F27 EF978EA8 EB932D76 06000013 21B078F2 30F15C01
F1CC0 - 30018F13 DB0137D7 C21351CF 8EED61DF 135077A8 9071088E 34151280 67D79061

F1D00 - 188ECD2F 4617F9F8 E86BE490 96F41258 820080F0 88060D30 380F0020 20307226
F1D40 - 5C880170 6044F31B F8458E42 FE44E883 8D20DA86 56196C90 720F5613 1F30EF6A
F1D80 - F2AE68EB CF470198 D612F080 17D0041A 31CF8555 CA700F77 E894AB27 CC94A28E
F1DC0 - 2522432D 2302DD28 CE4518B5 0181617F 8C93EE26 6C188017 7BF5B088 9FE647F0

F1E00 - 8F2F2C65 50850C65 50851C6C 6C655085 7C655085 2C6C6C6C 6550854C 6550856C
F1E40 - 65508550 BDA8EE73 1865A031 800EFE60 2F88227D 800EF6D3 AF2D68EB DE48D832
F1E80 - F088227D 600EFE6F DF88227D 50D8FE0E F6DCFCDB 0EF60EF8 6DBF8117 CC84747B
F1EC0 - 60564FC6 6AF88227 420D1E5C C4D0C558 F28640F0 EF5D08AA 7DE452D8 FC8CB004

F1F00 - 5D02662E E2859FAF F781053F AFEAFFAF E780053E AFB038D3 22B111AD 5731770C
F1F40 - 35808CF9 5160728B C50AC550 8EBBE443 196FD22F 30594302 258CD451 49F7342A
F1F80 - F28E3CE1 8C6F7FAC 27D228E0 BE18E134 116114A3 1DF96651 1F078F2D 214D8D64
F1FC0 - 4B18FD6D 31460651 77C5255A 78035808 C8F80845 84494CB0 873607F3 678F14A4

F2000 - 864C1704 68E3A141 7F13779A 11351711 337A26D6 133EE716 4C4E24A0 133CA133
F2040 - 75066410 76604B8A F41CF151 7865606B 01740176 E5727113 61338EF7 218BE51C
F2080 - ECE8B201 13117179 216E3F77 914606DD E7D11135 8548456A 4F31D01C 114D7985
F20C0 - 874808E7 E04171AF 214781ED 517D846C D5218648 07EC002A F10314B8 E09DE502

F2100 - 31E29627 185631D2 96250846 17153CAF 18418421 1810A8F8 1D40048E 08218F97
F2140 - 2618FFA1 618E5921 11A108AF 8AC18669 005A4D04 037D4011 011B8ECB DED68E4B
F2180 - DE10C153 78F8F5F0 11C8E4BD E1088EBA DE10B77A 48FBD3B1 7700AF46 7178D8BB
F21C0 - 811FE95F 21431C41 331C6133 14113303 1BE95F21 46134186 14601136 1B088F21

F2200 - 564136A4 E400A4E4 00A4E011 BF69F215 24AC2154 40100000 08FD6D31 5001617D
F2240 - 808EFBE1 8F85E318 F12E318F C0741873 70861918 D6CC7100 00000000 0000007E
F2280 - C38FBD3B 17E2F785 61C615F6 1B078F21 4A908E13 2FFF8EA3 A1D78E3C 5E407DB1
F22C0 - 55379831 37135028 F53AF08D 1AD31856 66008468 458471BD 79F214A1 1972EE10

F2300 - 9AC01023 19E8E876 E5B0112B 44102119 D78E555E 560664C7 3F1DC122 7DB54627
F2340 - B8147E94 8A187514 876837C9 E4137231 85587592 AC081181 1AE88158 1533C04F
```

```
F2380 - 7C5146A7 181D0CC9 6B808E69 9E8A8E0D 68EE6837 C418A8F7 8EF83443 5CC876E3
F23C0 - 86742122 96661814 814AE681 081096E2 01225218 7421CD45 11C114DB F6F60D5A

F2400 - B49A8544 4F894628 EBB3E886 B094C016 A7F88871 8754F206 330D3718 0240280F
F2440 - 0D57370D 980D0881 9080D454 02780F02 20273504 00811811 87400875 51876018
F2480 - 5314F965 80171843 03061371 F688F2D0 15B31370 70100000 00000000 00077934
F24C0 - 064A396B 60781020 33004F76 0033804F 8CFC6420 33F54161 FF2530E8 CC564D27

F2500 - FEF40031 A0DA3310 4F7ACF40 033005FA E66DBF85 4876031C F1CFAF21 378ED3AE
F2540 - 14313517 F17FE240 081E111D A1018440 38F73B90 161AFA35 FEFF4115 A5972400
F2580 - 01188E3C 9E135171 738CD11C 114B316F 9E26068B 03F5444A 514A235D 4E22F8F8
F25C0 - 90B14606 B318F3E3 20858910 D6811D2A 12D7A134 F3105C31 84FE1B4C F13D091F

F2600 - ED811F49 10F0E04F 640243A1 F2B8125E 01E5C6FE E800005B 28D989B1 31D014B9
F2640 - 66001710 38DE2C41 8C439E76 E16B3F7E D171BB14 B316FB46 96231E69 62F05F08
F2680 - DFF0C1B4 6B461544 7BEF5641 33713C13 18F7F2C1 7C9B1B19 8F214210 31448E12
F26C0 - C0174147 0A8F050C 11351C76 CBE96650 56076FB6 09871517 C8F57E8F 040C1135

F2700 - 56074006 E8E8F643 C1048900 02017315 D31C3012 C0D0D39C 2E200C20 08EE08E1
F2740 - 09715111 98E7E7ED 5D0E4857 7B415AA7 4311C914 75A07721 779FD585 67D11598
F2780 - 8F771C15 5F240D8F B41C1205 5E72A08D 36CB1779 0703B560 6A7B6FDD E579D014
F27C0 - 674006AC D135D014 B8FB13B1 041CD61B 0701A33C 2E2D5857 4508471B D79F214A

F2800 - 84584670 A0146135 8558EFC7 F733E7E5 E46062CE 1B198F21 46137728 97290578
F2840 - 73507FC9 A4C4001B 698F2146 13576994 41A4C4A0 7DDD6620 66F9844A 4CA4C42F
F2880 - 7E287D19 74CD1CF1 51766D88 46847D48 E2C4EDA8 55137108 8E65A08A 861783A4
F28C0 - 911B698F 2137144D 11181350 3874606C 4A8DD449 08DD5591 AC98C623 E8C5F2E7

F2900 - 6FF50030 9226F1AA 14509425 072D4762 03504103 F7ACBD9F 2F231142 4313F76B
F2940 - B8C546F8 E089072A F4F58E5C 901618EE 9614D48C 19907F74 73DF6E60 000AC214
F2980 - AA80A0CB 644008EE 6614D1D4 CC441D23 038BEA0A 86816022 86EA0605 50EBC408
F29C0 - 6DF01FD8 6F213614 56F6FD43 50D61506 C8F3300A FAE970FA 60EF4845 0A715014

F2A00 - A311F966 A0D3AF25 B08E1044 4748EC5E D4E377DE 96B418ED C2E4C28E 803E4323
F2A40 - 500000C7 CAA8E532 E4E08900 62030222 66D20000 093450C2 15016171 0F1F976F
F2A80 - 21431311 7714B310 E962728E 3C1143C8 EF71E4AB 350000F0 734A4BA6 98EAC58E
F2AC0 - C41E4B93 310F07F0 A4E8AC98 CCAEE8E7 790AC1AC 9760E421 3300AF76 E9B4558E

F2B00 - 8E88906F 20AC1AC9 7FDD4120 B87480B4 55BE1FD8 6F21478A EB085C21 0D0086D4
F2B40 - F1F244F2 14790E5E 07D507DA 0706DE06 DD061311 4B310F96 62C13706 3300AF7E
F2B80 - 591FD86F 21470885 58D80080 1108EB73 E31FF966 E8AC17F3 D4E4D575 16454873
F2BC0 - 80B4557E D90B8683 F7B7A087 B49D8AF2 CEAFA8EA 2037809D 92E31A0A F58EBA7F

F2C00 - 454713A1 338E463E AF2147E2 81EAF5AF 2A6E8BD4 0D5C5071 0A321188 FD791111
F2C40 - A0647084 00313716 21421F76 9F281C14 9171BF23 0615D57E D9162A6C 4111C114
F2C80 - F14C1615 DE850030 00000039 050D2A40 7B417526 D38E3DBD 451724C4 E535FFFF
F2CC0 - FE75B567 707D51AF 6D08EFFE 35903200 FD5F2F2A E973764C 414A311F 96604161

F2D00 - 78717716 5177386D 1AE5CF4A 214B1C10 EF0F6F6B 560EFA8E D7E356D6 98748348
F2D40 - F7AD514A 161311F9 62EA312E 966606C6 F74958EB BA34BC8C BDADD075 A57416D1
F2D80 - D5F6F60E F00E3AB5 6B26A2E5 D096A21A A2B568E6 0E3629F1 FA59F215 F6D0A8A8
F2DC0 - 1CBF6AF5 A0C461AE 9BF5F58E 76D35BE4 B8D2A6E7 675655F1 4A20312F AC296600

F2E00 - 1618EAF1 14D1161A 6D421312 09E190D9 81603286 996AF0AF 21811611 4A8ED20E
F2E40 - 5B08EA30 E4E08148 14B465ED AE094A00 80DF0D81 08100D57 FA46A4EA CA038E58
F2E80 - 21AE6B06 A6696AC2 AB605B36 A3E04932 D0266C16 2849F8EF 51144FD4 03BF4BF4
F2EC0 - AF2D681E D717D137 C213502D 1DB10832 F088FD79 11118D7A F227308A F5AC98E2

F2F00 - 7D0491AF 915C88E3 1937ED2B 455FD706 32036FFF 1F3015D6 17636FFF 1F2015D6
F2F40 - D21DCA15 D07C3330 715D01ED 79F31A01 4D210D00 80E834F1 8240B861 200B4011
```

```
F2F80 - F244F230 1155064D F32F0871 90460763 F75E2157 20B84B0B 15528E73 F01B097F
F2FC0 - 27850166 71502032 11874503 20187B40 7D9215F6 8EEFC04A 27E92147 8AEE1750

F3000 - 06360007 D5135147 C97D7214 5684F146 80D08840 0F68DFF8 1177501A CA7F1562
F3040 - A26454DB 13431BE8 E439D433 AC1AC98E 11C04428 E7BBD451 0A860E02 031038EA
F3080 - 2B3B4554 D68DE137 8E0AED8F 231218EA AED1351B 1B7F2307 15C01AD8 7F780074
F30C0 - 00700015 E623B064 01B264A0 D2CE15C2 16620031 FCA7F215 720B87B2 00B45178

F3100 - 8115720B 8580B155 2605E65B 080E8346 F8240B86 1200B47E 0774231F 9A7F2147
F3140 - F2D58168 E82B0486 7B708739 0E5D956E 7A703310 208E11A3 8EDB635B 0894A3F6
F3180 - 200B8648 085C5CC8 687C86A2 C315D8E6 E7D55B1F 344F2321 FF1553D9 F61F9A7F
F31C0 - 215D275C 206669D0 B16714E1 870B0116 61564186 A46500AF 215C8203 3237F8EE

F3200 - A9340035 02103F79 60400351 2303F7A5 04003501 603F7B40 4003D840 57313D0A
F3240 - 000AFAB4 43511003 F816816A E6812AC6 81274104 002EB94B 942096CF C018C8D8
F3280 - 31FD87F2 011F1B7F 2011F476 F2010613 61BEB6F2 0B15C20B 13407010 61361BEB
F32C0 - 6F20B15E 264EF061 361B198F 21441360 70106137 1F698F21 45137070 1061361B

F3300 - 198F2146 63DF0613 71F698F2 14766DF0 6136061B 198F2146 13607136 061361B1
F3340 - 98F21440 7629F061 361B178F 215076E7 F1360613 61B188F2 15471360 71360106
F3380 - 1361B178 F215276D 4F136061 B188F215 6761DF06 1361BBB8 F2144136 07010613
F33C0 - 71F0C8F2 14513707 01061361 BBB8F214 663DF061 371F0C8F 214766DF 06137061

F3400 - F0C8F214 71370713 7061371F 0C8F2145 07137070 11360613 61BBA8F2 15471360
F3440 - 71360113 6061BBA8 F2156765 EF2B1B0F 7F214416 40915C21 6207D507 1441640C
F3480 - 55FD906D B144032B 1B118F21 46D707D5 18414606 0C55F182 1460AD90 61841460
F34C0 - 370208C9 65E85480 F0207D00 84A1368D 80F20890 23891D28 9282884D 17AF68E2

F3500 - C6D4017E 11570884 A02480C0 2380C122 31FF2002 707D867E 18EF4AD8 F83DB013
F3540 - 7D7C2DF1 375D18ED 31E14313 014A314C 96650161 8EB39342 7AD396FC 02F30594
F3580 - 7A62E308 10AAF910 B5128E75 D04A0233 04551882 1580F088 F4423308 A872F308
F35C0 - 20AC7114 8EA39D12 072EC79F D8588210 389B5187 80178CC2 10D0080F 06BEE11A

F3600 - 80DE8888 E8E6EC11 1BAF511A 8ED28082 15006C9E 8CCF1320 8DE88111 BD87F215
F3640 - E676A656 066831A1 B7F0B156 20B868BE D794E328 7BF086AC 18697151 1759574F
F3680 - 546068A0 85BD6068 E8E1D424 86B738E1 F5D44384 A8492130 1902C430 29068085
F36C0 - 95012031 03966508 5A8E046D 47285B1A 1B7F0B15 420B19D8 DB15C207 DA206030

F3700 - 07DA1B09 7F215620 B85B0BF2 182AB2A3 E15C3655 260A21BB 74F215E0 A0E4B487
F3740 - 98E90A90 8466E528 448F3E32 0346C044 BD005890 F4390E49 90B42113 08C04020
F3780 - 1006DF19 68F931A0 9625B30D 879B0962 8A605196 6087BF34 8919E731 F516114A
F37C0 - 96880A6E 51FAF013 234084F2 135EA81C 73148E5C 132031D0 DA4607BC 36181854

F3800 - 7BD16671 7BE335B1 15B18ED7 234ED6A2 F75A314E 96AC0845 84671C26 29185551
F3840 - F8457283 DA846875 E0747314 E96AC085 0769257E 31308654 0E67B631 48DA6D41
F3880 - 85550C8F 9B2204D0 13014E96 EB178537 D134C031 A47613D4 6ACOAF2D BEE81EE6
F38C0 - DAD78E49 6DAF58EB 413451DB DA7D03AF 5C48EB31 331B4DA6 B8031B19 62F03180

F3900 - 966A0653 F60C01B5 74F215E2 0A8F6B02 01A874F1 5E21A1B7 F15620A1 AE74F14E
F3940 - D531F59E 502877BB 73A23500 B144AE68 E2313651 0867B384 47A60856 5345E440
F3980 - 17272773 28465128 4884B1B1 B7F20B15 420B4528 567942D6 70124AD8 66017AE1
F39C0 - 4908457E 218E7AEC 1B874F21 5E20A038 F9B2205C 08677086 4A4AF2DB 13584613

F3A00 - 014E96A5 08561371 35EE81ED A7AD187A 63876A1D 0864907E 81480D97 981D4038
F3A40 - 67417871 42F31257 17147E1C 114F1719 6E40D087 4A11C196 A9087640 E4D97441
F3A80 - 4AB8E62F 241B864D 031027B2 141A8668 187A3186 7E071113 1E47D01A F0133340
F3AC0 - 84F2EA81 C1FE74F2 D214FEAC 4DC81481 4D47401A F58E43F2 81081001 84073E01

F3B00 - 4ED7D814 A709031F 59E6C614 8D48FABB 10D815A0 0E0672B0 90C3DD01 4ADB8705
F3B40 - 014CB6A3 7B134B13 45907390 BE8AFDD7 86690CC8 A821C48E AAE25508 48DBDA03
```

```
F3B80 - 86611DB1 4CD0A6CC C52CDB14 CD402865 60CC01E4 011B874F 215E20B8 75200B01
F3BC0 - 31B18CD7 F27410D2 14E16113 2CACA132 031BE74F 20137B14 4B144011 B9A7F214

F3C00 - 6F280F42 280F4134 0320310E 0E6FB66D B570F649 0BB6C6C6 80D280CF 200372DF
F3C40 - 1BCA7F21 5620B2C8 7B200B4A 6D215421 B1B7F215 620B84B0 B154280D FD679748
F3C80 - 0CF13713 42031DF8 FCA90154 2173248A 8A1147A0 E4521791 32189132 55E200D2
F3CC0 - 91361357 424DA01A 4E59D1F9 A7F215D2 F2302210 D57DAC22 096A00B2 6A2E5D0B

F3D00 - 36A3E4F0 020B87B2 00B0123B 06A0E204 00BF692E 606470BF 680D180C FA46AC22
F3D40 - 040080D1 89471F6F 6D50D80F 3A4E31F1 03754056 31471721 537AFC81 6F2F280F
F3D80 - F1371DF3 892601DF 5A4E2007 1370302B F6A4EF6F 603F6F62 0D507137 0613706D
F3DC0 - 9AF86943 DBACBA4E 441A4E4E 2A4E464A 4E4047F0 3D2BF6AB B2F30420 94750A2E

F3E00 - AC203AD2 C6C672E2 DBF6AE9B 468E911D 5DCD2AAB C6C6F223 A8BBF2AB B2F304A4
F3E40 - B80FF80F 30394E92 15B615D6 AF680D38 94202050 075828EC BCD20037 772ACB8E
F3E80 - 3C0D1371 0BAF910A 8E897F42 5D5D2313 1DD8FD79 115E311B 1357832D 78E490D1
F3EC0 - 52316515 C216211A 1547AB67 02223304 20ABB6A6 F2B8C4D5 F1B097F2 73001661

F3F00 - 562B26A2 E4D00B84 B0B15420 384714A2 0312E962 83161314 C962C218 174C174D
F3F40 - 18F13DB0 857137D7 C28EC10D 145DF135 03877711 4A21B04A 0C204001 610314B1
F3F80 - 378BF201 37400171 03877A11 4A21B04A 0C400181 14A20031 378BF201 374001C1
F3FC0 - 03877701 81031C10 37C217C3 175E0309 98A60260 28EE3FD5 D02490C6 02003280

F4000 - 20820877 E073F073 017CA030 9986C18E 90FD571D 1AEC8ACD 032F1001 26022802
F4040 - 20877E07 6B076C07 F60AF6AF 779BF400 D9AFFAFA 94C35326 009B694A 86B8E80D
F4080 - 0BD00C5A FD0BF094 870BF4E4 7A7F4003 1F19E181 9EB3196B E0D9F2C6 0EFF0328
F40C0 - 02AC2450 B468E3AE C1471371 7F137145 1351CF15 37A4E018 168CC5EC 8128CC3E

F4100 - C8E591F8 D681F08D AB8118C0 A1F13776 DF203201 871EFAFA AF204490 2F0C4F12
F4140 - 0D0E431F 19EEE015 F3173975 BED68EAB DC131017 2A074514 055401DD 4143535D
F4180 - 454D4F1D 052594E4 455425F2 D4494350 5C41495F 37740594 F4049D4F 44454D41
F41C0 - 49253523 33232478 40594243 4D94E445 25643454 F4D94E43 54525D44 5F5D7425

F4200 - 14058494 34F60071 3706AF21 4F17080D 08901215 71171137 80913797 5BD1C1D2
F4240 - 14FD5071 3520D231 F1011FE9 5F214320 D231E3EA 1311C81C F863801C 81CF1537
F4280 - 17F14710 81731470 7037DA54 00203510 00098EBA 824007D8 540080D1 88040038
F42C0 - 910055D7 2B740024 7367400D 6F6F6709 7400D677 8740065A F8ED99C4 00310196

F4300 - 64003D63 0F2202DB F2C6C601 D0B24AF1 8C207279 6F561881 0080F089 7DE80F00
F4340 - 2118D271 57822C6F 683240E6 AF52771F 63520000 88E6E724 0078C455 17C61400
F4380 - 2034FF10 0571DAAA 0F0F075A 4400AEAD 68ADB0D5 E5F581DC EE942111 0D08E74B
F43C0 - C8BA6028 021108E5 3BC23A94 78C61007 A9640025 7B464007 0AE400D0 7BCE4007

F4400 - A7640078 26400203 108227E5 6400AF11 188AE808 E13BC2BA 95AF98E3 19C400D2
F4440 - 23713640 03120702 64003101 26791640 01187336 DAF6F67F F5D6227D F5400301
F4480 - 2371F540 07F9573B 34002678 B540035C 00008AFA 8E7A6240 07983534 7D20400A
F44C0 - F2155717 F15D7E61 7315D01C 515D01C7 E615D05B 180FF886 2080FF01 08E73224

F4500 - 00762573 75400D23 0CDA7645 40013676 75DB10A7 F7512AD7 75551341 128E8C9C
F4540 - 268EA89C 400D2316 DDA76CD4 007ABD40 07DA48E9 DA040071 2D400740 54002F75
F4580 - B4400277 CA44008C 1D2C7BFC 7B2D4007 A9440024 76A44002 17D94400 35001008
F45C0 - 77A44002 47584400 69CC7BBC 7BEC4007 A944007B 44400D0B 24786440 02F78344

F4600 - 00609C11 BD279841 0C8EC59C 1471C4AF 0143131E 2DA81CF4 F42B8A80 011C7554
F4640 - 8AE4003E 2560CAD2 744412B7 A34D68E7 E8C12B7E 2410C11A D77D0440 01148EA9
F4680 - 8C8E865C 4127F7C4 A07EA160 10C07AA3 40076934 0011B74E 3F2F28EA 5517CA34
F46C0 - 008EAC8C 119D772B 34008E61 5C4137D2 C55111C7 E93113CA 76534007 2834007C

F4700 - 0C5907C2 34001138 EDE7C11C 7D63C276 7310CF0F 07133400 61EE8556 60084512
F4740 - 01017B33 40096B56 7ABB4277 AAB86500 70E04003 50200087 9B274F24 001101D1
```

```
F4780 - 0157717F D18A8519 7671111D 615F38A6 9017367A 075FF022 5028659F D231028E
F47C0 - 15415AA8 457E9040 017315F3 23B16415 A1E91AF1 11897671 1C315F31 73121912

F4800 - A412178A 2D55A1AF B747223A 1E91A217 572AF77B 8056A022 03062102 8C9BBC8C
F4840 - D1028CAE 4C121173 74EF431D A7BDF4A0 1CF1C303 28027860 400D4814 AD0784A4
F4880 - 0077B140 0948927B E1400237 C9140081 0D6F2C67 6C140072 9F40071E 94003502
F48C0 - 00087F91 4007D511 53717F03 D079E940 07851400 20358100 08717140 07F21228

F4900 - ED95C330 00823916 F117BAF0 248E285C 958112C8 0F021022 345F8ECD 5C17715B
F4940 - 31738ACE D228E155 C7F41D21 5F38AEA0 320025C3 35C00008 73F04001 E119F77B
F4980 - E491DA1C F70604D0 78907550 5B0D223A 1EDA8E55 5CAF2AB6 F2D58E37 5CAB68EF
F49C0 - 55C23A12 8ED35C79 C0A99AF5 AF6BF2BF 22CDBAF3 A9703774 E400131D 01371C04

F4A00 - 90CA57F0 22490C00 037610AF 2A7E2315 5717F0D5 6F031F10 9F201228 C1812227
F4A40 - F0040063 586C7864 488C5712 8C751284 88CC4F17 1CF8C56C 18C0D028 C174C8C9
F4A80 - 92C8C8FD B8168168 C2C4C812 8128128C C94C8C97 4C8DBD03 1D9230B8 530BB160
F4AC0 - B863115E 023A1E0B 87320843 0B200171 AF40096B 9070284B 0119A4E1 09119B46

F4B00 - 590AF160 827ACD5A 18810080 F0887208 0F040053 EAD07A1F 400760F4 00203502
F4B40 - 00087F1F 4001D511 5F323B16 56066229 4BB1AF98 E0C3C91A D058E675 16F5115F
F4B80 - 391A1F1C 315B31CF 15771209 76201204 9D11C72F E912505A C2030E86 8921D51A
F4BC0 - F910B792 5DA8EFD0 15418E11 115B0203 002102DB 1FD19F27 10577AE3 23088EC0

F4C00 - 5F798E8E D2117C1E 15771081 1BAF5111 1D527CC4 F225B929 9A606680 1CB75B47
F4C40 - 85E23A99 27A957B3 E2BA9910 B1121D93 15971197 12E1D517 89411910 ADB791E1
F4C80 - 0C742EAF 511C79FD D77F871F 939F215F 712A1091 1BAFD1C2 7EF24006 E61D98EA
F4CC0 - 62C27A99 2BA952EB 074CF608 094F9F2E 90BD31DD 17704790 04E2B475 E5AF473A

F4D00 - D23B1211 1822BF4B F483240E 49960003 2EA831DD 177C3DA1 737EB323 C27D5DA9
F4D40 - 927A9579 6D7E3D2B A99AF577 5DD54606 ADDAFB7F 1DCE2391 AC0712DA F754E2DB
F4D80 - 0794BD07 2E040067 902E90FA 12D90B31 2030F210 22030146 FAFB8E67 1C7C3F4A
F4DC0 - E78A0400 75EC562A FB73BCCE 2391A902 D90BF02E A0F4A390 B53AF971 AC75BCDA

F4E00 - 788CAF58 14AD0793 C400756C 40073FB7 7F140075 3240020A F910BDB1 0A8F8912
F4E40 - 170E511A D711B97A 808CA17F 96FC0313 08CA4D18 C8F9BAF4 8E680C23 91C606E6
F4E80 - 0AF9731C 9122F814 A8073BB4 007FDB40 08ECCD04 00778B40 023708B4 00810F0C
F4EC0 - 431A0A62 75AB4007 46B400D2 759B4003 300217F6 B4007C3B 11015171 7F119155

F4F00 - 717FAF91 55717F13 6145174A FB708B15 D7748BAF 711C7B6B 1F119F21 53710115
F4F40 - D31737B5 BAF67BB1 AF215D71 771577AF 58EFEFB1 C37D91AF 215D3173 11996A61
F4F80 - B26550B5 697D80AE 21097371 17714714 71341741 5F770EAA FF1CC2A7 ADA14D17
F4FC0 - 18E87FB0 C51F2233 10087331 11A15D79 7D0196B6 6709A5F5 021F529F 275F010A

F5000 - AF9D2759 AAFA7A3A 400766A4 008E35C0 4007E0A4 0023770A 400810AE 6F2C6703
F5040 - A400207D E94007BD 9D22031F 1DA61601 192591E4 003762A8 AEF02590 A0FB064A
F5080 - E1359790 2AF97CF9 DA74B940 070E9400 71994001 11CC948E 0B444808 C0C817D9
F50C0 - 94008E25 714000B8 62200B4D E3300211 4F727940 0979296F 59D221BF 2BF214F1

F5100 - 710D51F2 00315D37 97914D17 30311B13 51121CF1 378B6421 3510B3F0 202A3C49
F5140 - 40584023 32413155 7007D265 007DC24E 28E066F4 527143D2 BF223A99 20109701
F5180 - 01088EE9 A0500625 0AF23088 E4CA08ED 8DBABB78 0323A962 00376D55 007D6241
F51C0 - 296BB08E 521F5606 DE2AF011 A8AE8028 6344DA1F 688F2D21 4F1371C4 13717815

F5200 - 7494E01C A102E481 C6860A46 48FA465B 0AF210A4 25A46542 AF2AE6F2 F2814814
F5240 - AE610A8E FBCB81C4 A2AF68E5 FCB1138A C50B2423 A9620714 212A8E8A BB1C3AF2
F5280 - 15F38E9A CB25A961 73157410 9AF21CD1 5F38E7AC B10C1CF1 57710884 87D38500
F52C0 - 63637CC0 5007C514 C27B318E 8B2F4F1A FB8EF5CB 06797B4D 07D008D1 75108C3C

F5300 - 1E7B2116 AD215C06 F647D705 007D014D D7CE08E9 A2F60BF7 16050071 F041C70D
F5340 - 02C90BD0 208E482F 4BA207E5 078C0E48 EF22F469 769F16E1 6D14681E 816816E6
```

```
F5380 - 812812A7 61447590 6E4F7990 16B15647 DE350007 D7078EE7 BBAFF067 67016A2C
F53C0 - 1560A872 01881468 E93DE4B0 30C226E1 F16716E1 5E68EBF8 ED718315 A316316E

F5400 - AF214681 EF6F6CA0 27C1016C 16BDB154 37ECF8C0 D7E8C054 B8D5CA31 D214E80D
F5440 - 18927088 3848EB6F D1F388F2 07715007 8E6DAB09 8EBBFD8E A3BB8ECC FD0A8E1D
F5480 - AB068E3C AB068316 06B20253 4A88F220 1B278F2D 615C9038 128C58AB 87080210
F54C0 - D008727F DBB06442 DBA065C1 90E2E11A D5B06480 A0545162 B211AA06 90E2C6A2

F5500 - 190D9B87 14BF78E1 22F56063 928ED2AB D674B6DB 10B7B774 606CB213 5A4DAC2B
F5540 - 46B49560 699717F1 5B37C371 5938E888 0119816B 46440D71 208E7A9B 1047417A
F5580 - F215F311 37F07150 78E589B7 41FD6F2B F21091D9 315F710A DB8E17ED 8488ED15
F55C0 - F4268EC7 ED1097BC 6156710A 7B86D511 38EB39BF 411AD610 B8EF00F4 C211AD78

F5600 - E90DE521 8EE26E45 173164E0 119937FD 6E516296 8178F77F 90339300 40017F17
F5640 - 314B17B3 020E0290 A606E86F 4D2A86AF 01432580 F0EA80F0 2090A42A 064A1A06
F5680 - 5606580A 06560658 06FA0672 E1021013 05133131 CA8E858B 1CF15B38 4033412E
F56C0 - 23916508 508E238B 119E681E DA101860 528EA18B 20D23113 EA11A10B 8FC1C701

F5700 - 1B10A6A5 0AF210A6 C8F101AF 2D697260 68B08E32 8BAB6F68 EA08B10A D2305646
F5740 - F308EA10 117415B7 1021111C 430D654F 72F48171 208EFA7B 1048588E 263F4368
F5780 - 21038E48 BE501ACB A4657094 A00006F1 D8178EA8 FE067FFC D679FC10 C11AD781
F57C0 - 70750288 071D0102 30621614 32030C21 634E72C4 40176732 033F3000 2D810113

F5800 - 51574A4D 114ACOB4 4BCC0E40 1041F939 F294A17A 465606AA 0A464C0A 46454682
F5840 - 0231D12A F278A8BF 2BF210AD 297AC773 03667FAF 017315B3 1031C315 B31025A5
F5880 - AF010314 BF0F0171 14B56E13 7172DAAF 214F1311 5B525B1A 20315025 A1A10290
F58C0 - CD911C7D 93BF2BF2 A7699688 78538EC7 6B37B656 97379762 134C02E0 8A1606DE

F5900 - E779311A 154716F1 2311CD61 13154716 F1191448 F77F9014 610919BA 156710CA
F5940 - F0DA1031 9B915671 0A4A07C0 26E3473D 20775331 44164091 5C28F1C6 117E1314
F5980 - 2DE06190 A15E20A5 7115837B C11B0A8F 215E3021 11D23141 CA13311C 15541FF8
F59C0 - 8F215741 3111280D FD288190 20308EA2 0305EA82 225A8081 C2082183 2607EBA1

F5A00 - 02843705 2817751A 875B1114 8EEF4B7C 024738E2 10F4E211 1D23152C AD21FF88
F5A40 - F2157494 A32A465E 01C814F5 414C1A46 4B0A4645 0308CA13 18E75AC4 A61128A8
F5A80 - 117F81D6 8E44C043 5831227B 71D2E6DA 8EFA014B 38EEAD04 2315D086 52220D07
F5AC0 - 4508E721 B4D096B8 08EA8DA6 80179415 7F81680F ED610B72 41119135 17FD215D

F5B00 - 31CF8F64 1A011BDA 80DE8128 0C18E809 DD8112E0 83140E4D 8F4F411C 7B21E243
F5B40 - 1D0B608A C50B2473 208EEE8D 80D18900 062CA7CB 8D0B2411 C72F0865 44CE7A50
F5B80 - 10C4837B 987780D6 8EFBF045 28A8DC8E 1AB05828 80218EE1 CA8985D8 960D208E
F5BC0 - C20B5008 C439CCC4 CE0D58F4 1C8CE63B 11110220 D23101CA 131D015B 333802E8

F5C00 - A6F08FFD D0111A10 96D6B253 088E0DFA 50030E03 8C449E8C 796E7E10 1B109F29
F5C40 - 7C601527 817B4740 0A4F0385 38C9F5ED 28C194F8 CAC2B8CC CAE26741 04008E11
F5C80 - 6E4008C1 90B228CB 2F01F519 F2011BB9 8F201D91 0B71BFDA 8D950118 CC28D73C
F5CC0 - A50075AF 4EE8E2D4 F10A7DCF 5737C005 038E5FEE 4ECA4DAC 980DF891 00893000

F5D00 - 311B816A D28CDF2F 7F7FD215 D373EF48 9730F419 84832308 8E8D3E11 A2614B96
F5D40 - 87217C17 715B6912 E017E17F 17A5FD1C 7AF215D5 20641A72 1A500707 070FE5A1
F5D80 - 881A180F 08862080 F0521470 30E21691 F6BF9843 70401201 018EF0AE 44E76EE7
F5DC0 - 6208E36C E151717F 11015937 62F41C76 4E4AB699 9853DB10 9746EAFB 129D7036

F5E00 - D807D795 007F5E40 F709E460 66D97BCE 0B80F00B 86BE0851 860416CA E86A8084
F5E40 - 05508500 B80F00B8 0CF13517 E15349CA 8B1C4173 A4E59F15 F57D1E8E F82F718E
F5E80 - 4C071AD8 21500692 E7FE8500 978627CC D4D7DB13 58E434D7 E737ED47 B47D069E
F5EC0 - 07E50B1C 3020202E 414D4540 20202023 50245950 55402020 2C454E40 20202024

F5F00 - 41445540 20202024 594D4540 2D0A0FF6 79DDB135 8E5B3D07 1358FE0C 108ECC3D
F5F40 - 137D78E2 39A43D8E 819E7162 42F8AE60 66808EE9 A05318EE E8A7E147 B864E05A
```

```
F5F80 - 78EAD8A4 438FEE01 08FAB251 4908FE21 20968CE8 EF53D137 D7D88E3C 8AD44951
F5FC0 - 1B10A8F3 E32033F5 023A603A D602AB90 0096841D 01FA49F2 15908AA4 18E57FA1

F6000 - 4379A0CA 141791C2 08210380 C1068FEE 0100780D 165FE772 244E8AED 3674F777
F6040 - 26FEF7F0 24CC8AE6 F11B94E6 17D56756 65B011BA 4E10B73B 244A6FEE 11A7B26D
F6080 - ACCCC47E 7E167C36 764BE670 26CC5FE7 1167906A C25CBD72 0D231050 3210D007
F60C0 - DF58F83D B08EBFBB 5018953E 898ED69D B8AB5D8E 502E4CC1 7F8E98C0 17F8EAB0

F6100 - C8E62EB5 33CC4361 018E457E 7D9040C8 AAE4111C C4521017 52149A8A EBE5432B
F6140 - 4C900000 00000001 F0A8F2DB 14576321 F0A8F214 7704F560 D220AF0D A7345BF0
F6180 - BF0A0C1C F137068E EFDA0713 38B62A13 31351517 8ED00C79 5E8DE83B 1400AFB7
F61C0 - 7E4F281E CE7CE4D9 760A10A1 0B8FBD41 08E01AD7 37172A08 0CEAC255 0B468E72

F6200 - 2D8F6A41 08E8E9D8 E132D80D E94E008A AF211B70 84728410 B7A7ED58 E04DA143
F6240 - 174147E9 8B670145 038C59CD 11A8AAE4 94A41715 479547CF 040011A8 AA22CE78
F6280 - 2472247C 3478B040 091A0EB1 654111BD 2AC2A4E1 0BD20311 A10BE603 11A79E3C
F62C0 - E8AAA470 E310A25D A90E217C E34B0D67 9E35F0D6 70E37D70 40011A72 D3784040

F6300 - 091A4B11 A10B0311 B7593CE8 AA297093 608F11B7 F832590A 61D57530 400D912B
F6340 - AC212BDA 7E88AC21 0A8148E3 35E4001F 519F215F 32303D0F 6ABA8E91 FD4007DA
F6380 - 8400228E D3804008 C90FD772 38E700D1 B109F215 6716F155 717F1461 6315D317
F63C0 - 38E79BA1 331312B1 5DB17B0C 56F13117 5AF27312 8E60DE79 02AFA77B 84A2AC3D

F6400 - 6F2C65A0 B4723B98 8EEC9A24 7732D117 15B3A4D1 C3AC9A46 BCA80DF3 70235055
F6440 - 414D173D 91371741 5B913715 9917B8AD F1167729 124AF28E 28CEAE27 0816590D
F6480 - 9E6134AF 21564161 14EAF51B 939F294E 4215A520 315025A1 2B19550A F2B7681E
F64C0 - 6050A465 50480A46 5C01B129 F24A8A46 59174118 E50CE7A0 1BF281E5 9115E316

F6500 - 3AF015A3 8E419AAF 6AFA8EAC 8A2F90D9 00D58F20 80CFAC5A C3203500 B4D42F30
F6540 - 598562BF 6F6BF5BF 505A05BF 5550B750 42C3086A DFDA2496 A400D7AD 09688014
F6580 - 91711711 B929F215 E516314C 183BF6F6 15C320AF 23103DA3 9F2F202A 30216015
F65C0 - A0149171 18015A01 61149171 14D171BF 6BF696E4 D2B1CF0C 5AF8E9DD C0313613

F6600 - 71360120 8EE86F15 2710072E F8F43581 77DF1108 EF66F150 7133D231 82DED78F
F6640 - AFD718FF D1518DE9 22090DF0 1710D55F 201C194B 511371DD 21371C11 4D17180C
F6680 - F80DFA89 890A0B96 0D55F213 0314D171 A4E5DD03 E68C1A8A D2E68C08 8A8C778A
F66C0 - 8C4C8A8C BE3E8CBF 3E25A9A7 A744008A 8267E504 62890941 32182132 4C015D51

F6700 - 755BD203 07220289 4008E7B0 A89121AF 68963B89 8EA59D80 D46221CC 14D1715F
F6740 - 90203167 0B15E00B 86062160 15E68161 88A46560 220380D2 88820200 18508518
F6780 - 61811601 5E01800B 870200B4 3286C521 361B244F 21564134 94AF0B46 49018724
F67C0 - 02861E08 60F16B7F 1670B15E 00B84186 0E952116 70B15E00 B8601116 015E6816

F6800 - 18803840 637F0B16 815E0188 0B012031 20690020 31607D43 400799F4 008E28F9
F6840 - 8810F80D 4BB28806 0F60380F 02202758 F4008E85 F988B208 0FF5006E 8E786F40
F6880 - 08EB3F98 865E52E8 EB94A551 022DA836 B10208EC 84A35800 00A2DA83 A0F76924
F68C0 - 00AF0AC3 797E487A EA814814 BF6F6B47 0D5CE308 9C7DD203 1D02D90F 50AE22F9

F6900 - 6AA09665 0A908148 140D0D58 E2D90B32 2FAC2978 6190C700 D58F80FF 81EB46AC
F6940 - 72003894 0080FF8E C7E9891B 08966066 ADACB80D F8903881 08100D52 F1FBA8F2
F6980 - AF415171 12F0F0AF 111980DF 89250A7D 207E7015 37AF8018 7CE11321 824F1132
F69C0 - 15F51757 CC053E02 7F204FD5 72162132 A6C4D614 F1717351 40086CAE 770043E2

F6A00 - 00286800 1361B244 F2156413 4A4E018E 3D1D87CC 21321854 D2132AF9 7B50400A
F6A40 - F9BF6BF6 7B4057D0 2037CAF4 1D54A165 132A6C4E 2AE973D0 400A6C4E 1D9F6F67
F6A80 - 0C040086 CAD747F4 3D57C038 0FF26318 116680DF 87CD10B1 5E00B871 01870B01
F6AC0 - 8615C703 850851BF 2BF2BF2B F2BF2136 1B244F21 56413416 115E0181 0B870200

F6B00 - BBF6BF6B F6BF6BF6 80FA5708 71C0890E 00C49018 6240280F A861B086 0C2617F6
F6B40 - 65080FF2 23500004 16F4F263 10120166 87C310B1 5E00B870 606C4F84 0615FF2F
```

```
F6B80 - 2F2BF226 31012016 687C210B 15E00B87 0505EC84 1642FF2F 2F2BF26E 9F80F021
F6BC0 - 32A416BE F80F0213 2C416CDF AF570000 71360613 220346A0 00CA1321 56780D0B

F6C00 - F6890279 11111641 36809136 5ED80FFD 1CD18315 E323A065 B0AE6AE1 473A0653
F6C40 - 2D520320 EF0EF1FE 0EF20EF9 320EFDD5 01A065A0 20A86A81 BED210D1 36071360
F6C80 - 10008744 14451400 28554E44 40068594 4495F340 555E4C40 244BC494 354554E4
F6CC0 - F540555E 44504447 4514C4B4 06445351 4440A445 4444C40C 44544444 50058525

F6D00 - 44950940 5946434B 9405C405 44104057 445C4404 05354434 0048534D 44420F05
F6D40 - D4C41440 F05D4451 400000AF A8E580AA F997A002 FBF20D94 A8FBF6AB 280F0200
F6D80 - 30520A81 A0D15379 80400304 B04B0496 C7117115 3717F100 15370502 268C507C
F6DC0 - E4F00368 00AC214A 311F9621 18E422D4 538EEABB 8E25EC46 2AF22730 815C8752

F6E00 - A4318EBE 3C4A08C1 79A2864A F8EAF0D4 00877D07 834D7555 028A8C48 407B564E
F6E40 - 331A3962 4131E296 23187072 727670F0 67007B91 4007A26D 74113102 96280250
F6E80 - 2D38ED15 C8588ABE E8E6E994 008CC2FC 8E370D40 077358EC 60A11C2B A9E10420
F6EC0 - 84886790 8506E6F8 E8A0A143 D23141D5 8E0A0A13 7E98B6A5 14513511 8155717F

F6F00 - 11C8E030 A15D3725 3400D779 6F400068 E160A153 717F1001 438E7EF9 10417313
F6F40 - 71450703 2B022502 754547F7 C954977C 65726245 E78A2460 6F90109A F910A781
F6F80 - 54A32031 82966B27 5F04007F F441B319 29668AD9 CE490219 0AC02802 7215D211
F6FC0 - 2AF8111F 2F2AE621 0D5D331A 296670D2 5F231529 66D07890 4F1608F7 3D47AF06

F7000 - 01071C14 00D231F5 400109AF 910A7974 4C531A39 66F47850 40031309 E190D9CE
F7040 - 56028021 12AF8111 816310E0 E6AB6681 2F2F24C0 C6C60E3A 03F2AB60 3715411A
F7080 - AF511903 AF17A044 05723440 1F1A8895 9AE28023 1E2966E2 75E34B27 D0440205
F70C0 - A04550B3 50447D77 C34D07FE 354F7EE3 AF48EE7C 9D1AED8A E2B31F10 3AF17993

F7100 - 44271C34 01F1A889 29AE2802 31E29626 079A3AF4 8E93C98E 22E9AF5D 1E57A534
F7140 - 34728348 3F1A8892 9AE72434 937A634E 205A0455 0B650445 A74234B1 7C435FE7
F7180 - B4392DB0 F158F7D3 3D0AE48E BCB9AE53 1F19E16C 8118119E 9BB9696B D2AEDF5A
F71C0 - 350E3D03 AF17AC24 E2712343 2AE88158 1596D317 EA242176 D256E4CD 0370D297

F7200 - D6025029 6DEE8158 1554F8E0 0FC5008E 34FC533A F237E455 C4C47220 31F75003
F7240 - 7C4F4F40 57D0031F 956031F3 0297520D 20172324 D020312E 962606E3 114A1613
F7280 - 14C96260 6470783F 4007E7F4 505E5722 174F1453 312E9668 27F1276E 18E35DC4
F72C0 - 00CD4E02 190D7020 5C028027 3F1D1133 8E620C13 38E3A0CA FDF2F2AE 66C50312

F7300 - F966217E BE400D23 1F552431 58966717 1B178718 E5ECC400 6C6F3138 96660D20
F7340 - 37981788 172518EC FCC40014 A06312F9 66200741 51617350 70617721 8E49CC40
F7380 - 031309E1 90D9CE56 0280210A 1378EB6F B1371098 E5EFB12A 669C2502 10AAF912
F73C0 - A0313706 1371F698 F215D307 135AF98C 18FBAF21 08867708 A8332F30 87E20AF4

F7400 - 1004F02F 3027F10D 40333020 2DA1188A A4003D02 502850AC A8E72B9A F5AC6785
F7440 - 04337FA0 5E087032 74704C18 40AE8815 815A4E94 E4DAA054 17950AA0 A2C86050
F7480 - AF1942E0 811811B4 651FB240 18C7CAC8 E10FB8FE 3F8007DF 06DB068E ACA98E01
F74C0 - FB8C3FDB 8C3C998C 2FAC07DA 07DE068E 2CEB8FB0 F808E1EE B8CE0CC8 CD6997F3

F7500 - 6317E966 366CA47E 9471308D 4A5307F8 472208F8 26304501 71312F71 85858849
F7540 - 8DB32307 5F531DF9 62D0312F 96240073 14F74556 3B527605 17756AF6 3594A554
F7580 - 97644175 7BC25808 C74FB831 80246121 7C958E13 905D07D0 572B44D0 626577A4
F75C0 - 53E6AF02 764F07B6 17EB50E7 D31E2D30 07635660 F7355AF6 35FEFF52 976F17EC

F7600 - 47E847E5 4205CA78 94311F64 A47DF485 884A6893 7EF47C95 AF637840 594C4976
F7640 - A21775BB 7745FEFF 62420007 0C47435F EFF42E00 00208D53 03018503 7715FEFF
F7680 - 62900006 2EF1857E A48588DB 7B207BBF 70E35F17 59431389 66A07C05 6204258C
F76C0 - E0EB6654 70707990 5BF86816 70514A57 8F757172 4154F1C1 14B9627F 5E37C147

F7700 - 724869A2 70C4AF63 554F4C49 76611757 1A4AEE25 7DA37AF3 75305798 4A037353
F7740 - 20312F71 73791342 1312F966 90706356 0754364C 37A8D400 7054AEE1 33101133
```

```
F7780 - AC281481 4B461711 4B716D5C E80DF0D8 108100D5 7FA46A4E 80DFA961 08D08E61
F77C0 - 4F4E5110 ACA80DF1 19809135 AF684A84 98489695 08582F36 534D4449 72513874

F7800 - 41445149 76808598 5AAC680D F7CB26C0 31811191 35027362 311F7382 7422870E
F7840 - 08737086 A4003795 275D2027 92246065 4120314C 7F422F30 A74B1453 10431A39
F7880 - 62D231E2 96602171 332E2F7B 222F3067 A8147094 CF662F01 7184A7B0 37382313
F78C0 - 89667125 86AC0207 8E164D06 6C031589 66A576D1 7B617B71 46E31829 662274C2

F7900 - 73517361 4EC31922 49663C20 742231A3 96651312 F7E81762 17631486 6B607BD1
F7940 - 7F417F01 7F114A28 315057C7 0C114B72 B155B1C1 14B17131 029623A7 F2120314
F7980 - C76312F3 087F904F 0948A076 A1685F21 0D007371 82111403 84A66008 5A8488FE
F79C0 - 7A205B18 78606DCB 77412031 1F7DD003 77317C90 4606F217 73131E29 622231A3

F7A00 - 96650171 79AE4B08 312162BB 667B7D6E 6BEF84A0 37DF020A C07BD050 087100A4
F7A40 - CA4E4007 3701717F B04EE031 3710B135 8D9DF307 CEF873B0 11B13559 08704003
F7A80 - 26027B00 74DF873B 10313713 58E79491 3613410A 0111A134 8E794913 501AEE8D
F7AC0 - BEC208DD FC20AFA8 D51D20AF A8D62450 773011B1 3713510B 7C6F873B 011B1355

F7B00 - 90870400 3268CEB9 B14B8D96 F301FFC6 F2143131 03203102 1C117114 B9627F01
F7B40 - 8DB39407 840FEFF6 29000060 7A1857BD B84A74CC 4606FAF6 99F7C10F EFF32900
F7B80 - 00671A18 584A8586 D2E8DA2C 2075FF0E 2101ED00 0067DF79 AA8FE7A2 0460702F
F7BC0 - 6A5F312E 62FE8D8B F3035943 5027DEE1 4B700357 17BB177E 25011717 ED153E7F

F7C00 - 6268507C D2572779 173C2171 9669078B 15BE79A1 14B8D054 507B326A DF3794A5
F7C40 - 54022718 1798E8F9 57507082 4C014B8D 30350171 7D617102 63EF77C0 310E962B
F7C80 - 0311E966 4E671215 B5AF635F EFF52976 728498F9 91507822 171962D0 1C172117
F7CC0 - EA1679F7 D0259017 1698F690 F31FE966 C01757DD 15607400 6A6F3394 F46ACD7B

F7D00 - C066A173 C064DE37 840594C4 2778BD6D 8F70A114 B72107D2 05BF7B40 54F6F1F3
F7D40 - 12F96600 722133B3 0274AF17 114B0374 81480027 35D17114 B8EC0195 FE7350AE
F7D80 - E03311F9 66007ED0 77415A07 E3060FF7 E20AEE03 31A37910 17114B31 38966421
F7DC0 - 8117131A 261FC732 F310263F F31C26BE F3158966 F2315277 DF78707C E0506318

F7E00 - 274CF756 0319278B FAEE5643 14C96642 17114BD6 A664C017 1798C5DE 7AA040C5
F7E40 - B1312F96 6E018131 E2DA59D7 610312F9 62400331 A37B5F17 18D22950 3F34F4E4
F7E80 - 4525F4C4 022F764C 74405901 7160CD60 4D8F2915 014F80D1 0C2045E6 64D77008
F7EC0 - D1055039 94E44525 0229610C 14B311F9 62000131 2E962000 10000000 00000000

F7F00 - 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
F7F40 - 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
F7F80 - 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
F7FC0 - 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00001300

/SLOAD:  End of Saturn Loader Execution
```

```
 1            *
 2            *       N   N  ZZZZZ   &     RRRR    SSS   TTTTT
 3            *       N   N      Z  & &    R   R  S   S    T
 4            *       NN  N     Z   & &    R   R  S        T
 5            *       N N N    Z     &     RRRR    SSS     T
 6            *       N  NN   Z     & & &  R R        S    T
 7            *       N   N  Z      &  &   R  R   S   S    T
 8            *       N   N  ZZZZZ   && &  R   R   SSS     T
 9            *
10            *
11                    TITLE  Rom start (header) <830927.1416>
12 F0000              ABS    #F0000         TI%HP6 address (fixed)
13 F0000      =ROMSTT
14 F0000              BSS    #8-((*)-(ROMSTT))  8 nibble rom ID
15 F0008              END
```

=ROMSTT Abs 983040 #F0000 - 13 14

Input Parameters

  Source file name is NZ&RST::MS

  Listing file name is NZ/RST:TI:ML::-1

  Object file name is NZ%RST:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1                      TITLE  Lexical Analyzer Tables--ID=FF
 2                *  This file was generated on Tue Jan 24, 1984  12:00 pm
 3                *  File Header
 4 00000 8405           NIBASC \HPILROM \       File Name
         94C4
         25F4
         D402
 5 00010 0000           CON(4) =fLEX            File Type
 6 00014 00             NIBHEX 00               Flags
 7 00016 0021           NIBHEX 0021             Time
 8 0001A 4210           NIBHEX 421048           Date
         48
 9 00020 0000           REL(5) =ChainE          File Length
         0
10                *
11 00025 FF             NIBHEX FF               Id
12 00027 10             CON(2)   1              Lowest Token
13 00029 62             CON(2)  38              Highest Token
14 0002B 0000           NIBHEX 00000            End of lex table chain
         0
15                *
16                *  Speed Table
17 00030 0              NIBHEX 0                 Speed table exists
18 00031 000            CON(3)    0               A
19 00034 F00            CON(3)   15               B
20 00037 450            CON(3)   84               C
21 0003A 270            CON(3)  114               D
22 0003D 2B0            CON(3)  178               E
23 00040 C12            CON(3) (TxTbEn)-(TxTbSt)  F
24 00043 C12            CON(3) (TxTbEn)-(TxTbSt)  G
25 00046 C12            CON(3) (TxTbEn)-(TxTbSt)  H
26 00049 EC0            CON(3)  206               I
27 0004C C12            CON(3) (TxTbEn)-(TxTbSt)  J
28 0004F C12            CON(3) (TxTbEn)-(TxTbSt)  K
29 00052 1F0            CON(3)  241               L
30 00055 C12            CON(3) (TxTbEn)-(TxTbSt)  M
31 00058 C12            CON(3) (TxTbEn)-(TxTbSt)  N
32 0005B A11            CON(3)  282               O
33 0005E 931            CON(3)  313               P
34 00061 C12            CON(3) (TxTbEn)-(TxTbSt)  Q
35 00064 171            CON(3)  369               R
36 00067 3D1            CON(3)  467               S
37 0006A B02            CON(3)  523               T
38 0006D C12            CON(3) (TxTbEn)-(TxTbSt)  U
39 00070 C12            CON(3) (TxTbEn)-(TxTbSt)  V
40 00073 C12            CON(3) (TxTbEn)-(TxTbSt)  W
41 00076 C12            CON(3) (TxTbEn)-(TxTbSt)  X
42 00079 C12            CON(3) (TxTbEn)-(TxTbSt)  Y
43 0007C C12            CON(3) (TxTbEn)-(TxTbSt)  Z
44 0007F 0              NIBHEX 0                Speed table exists
45 00080 4610           CON(4) (TxTbSt)+1-(*)   Offset to text table
46 00084 0000           REL(4) =PILMSG        Offset to message table
47 00088 0000           REL(5) =PILPOL          Offset to poll handler
         0
```

```
48                      STITLE M a i n  T a b l e
49                *  Main Table
50 0008D          =xromFF
51                *
52 0008D F00              CON(3)   15          01  BINAND ( <int.exp> , <int.exp>
53 00090 0000             REL(5) =BINAND
         0
54 00095 F                NIBHEX F
55                *
56 00096 E10              CON(3)   30          02  BINCMP ( <int.exp> )
57 00099 0000             REL(5) =BINCMP
         0
58 0009E F                NIBHEX F
59                *
60 0009F D20              CON(3)   45          03  BINEOR ( <int.exp> , <int.exp>
61 000A2 0000             REL(5) =BINEOR
         0
62 000A7 F                NIBHEX F
63                *
64 000A8 C30              CON(3)   60          04  BINIOR ( <int.exp> , <int.exp>
65 000AB 0000             REL(5) =BINIOR
         0
66 000B0 F                NIBHEX F
67                *
68 000B1 B40              CON(3)   75          05  BIT ( <int.exp> , <bit positio
69 000B4 0000             REL(5) =BIT
         0
70 000B9 F                NIBHEX F
71                *
72 000BA 270              CON(3)  114          06  A=DEVADDR(<device spec>)(Retur
73 000BD 0000             REL(5) =FIND
         0
74 000C2 F                NIBHEX F
75                *
76 000C3 290              CON(3)  146          07  A$=DEVID$(<device spec>)
77 000C6 0000             REL(5) =DEVID
         0
78 000CB F                NIBHEX F
79                *
80 000CC 380              CON(3)  131          08  A=DEVAID(<device spec>)
81 000CF 0000             REL(5) =DEVTYP
         0
82 000D4 F                NIBHEX F
83                *
84 000D5 ED1              CON(3)  478          09  SPOLL ( <device spec> )
85 000D8 0000             REL(5) =SPOLL
         0
86 000DD F                NIBHEX F
87                *
88 000DE 281              CON(3)  386          0A  READINTR {read interrupt cause
89 000E1 0000             REL(5) =READIN
         0
90 000E6 F                NIBHEX F
91                *
92 000E7 171              CON(3)  369          0B  READDDC {read last D.D. comman
```

```
 93 000EA 0000          REL(5) =READDC
          0
 94 000EF F             NIBHEX F
 95              *
 96 000F0 CF1           CON(3)  508            0C  STATUS [( <loop #> )]
 97 000F3 0000          REL(5) =STATUS
          0
 98 000F8 F             NIBHEX F
 99              *
100 000F9 EC0           CON(3)  206            0D  INITIALIZE [<volume>]<dev spec
101 000FC 0000          REL(5) =INITXQ
          0
102 00101 D             NIBHEX D
103             *
104 00102 450           CON(3)   84            0E  CLEAR LOOP[;<loop>] | <device
105 00105 0000          REL(5) =CLEAR
          0
106 0010A D             NIBHEX D
107             *
108 0010B 000           CON(3)    0            0F  ASSIGN IO
109 0010E 0000          REL(5) =ASGNIO
          0
110 00113 D             NIBHEX D
111             *
112 00114 A11           CON(3)  282            10  OFF IO
113 00117 0000          REL(5) =OFFIO
          0
114 0011C D             NIBHEX D
115             *
116 0011D 2C1           CON(3)  450            11  RESTORE IO
117 00120 0000          REL(5) =RESTIO
          0
118 00125 D             NIBHEX D
119             *
120 00126 1F0           CON(3)  241            12  LIST IO
121 00129 0000          REL(5) =LISTIO
          0
122 0012E D             NIBHEX D
123             *
124 0012F A21           CON(3)  298            13  OUTPUT <dev spec> [USING] <lis
125 00132 0000          REL(5) =OUTPUT
          0
126 00137 D             NIBHEX D
127             *
128 00138 1C0           CON(3)  193            14  ENTER <dev spec> [USING] <list
129 0013B 0000          REL(5) =ENTER
          0
130 00140 D             NIBHEX D
131             *
132 00141 321           CON(3)  291            15  ON INTR GOSUB/GOTO <line #>|<l
133 00144 0000          REL(5) =ONINTx
          0
134 00149 C             NIBHEX C
135             *
136 0014A 3D1           CON(3)  467            16  SEND [;<loop>] {<Frame>}+
```

```
137 0014D 0000          REL(5) =SEND
          0
138 00152 D             NIBHEX D
139               *
140 00153 5B1           CON(3)  437             17  RESET HPIL
141 00156 0000          REL(5) =RESET
          0
142 0015B D             NIBHEX D
143               *
144 0015C 061           CON(3)  352             18  PRINTER IS code
145 0015F 0000          REL(5) =PRNTIS
          0
146 00164 D             NIBHEX D
147               *
148 00165 1A0           CON(3)  161             19  DISPLAY IS code
149 00168 0000          REL(5) =DISPIS
          0
150 0016D D             NIBHEX D
151               *
152 0016E A41           CON(3)  330             1A  PACK <Device specifier> code
153 00171 0000          REL(5) =PACK
          0
154 00176 D             NIBHEX D
155               *
156 00177 931           CON(3)  313             1B  PACKDIR <Device specifier> cod
157 0017A 0000          REL(5) =PACKD
          0
158 0017F D             NIBHEX D
159               *
160 00180 4A1           CON(3)  420             1C  REQUEST [<loop #>;]<num|str ex
161 00183 0000          REL(5) =REQST
          0
162 00188 D             NIBHEX D
163               *
164 00189 CF0           CON(3)  252             1D  LOCAL [<dev.spec|loop #>],[<de
165 0018C 0000          REL(5) =LOCAL
          0
166 00191 D             NIBHEX D
167               *
168 00192 591           CON(3)  405             1E  REMOTE [<dev.spec>..] | [ LOOP
169 00195 0000          REL(5) =REMOTE
          0
170 0019A D             NIBHEX D
171               *
172 0019B B02           CON(3)  523             1F  TRIGGER [<dev.spec>..] | [ LOO
173 0019E 0000          REL(5) =TRIGER
          0
174 001A3 D             NIBHEX D
175               *
176 001A4 551           CON(3)  341             20  PASS CONTROL <dev. spec> | LOO
177 001A7 0000          REL(5) =PASS
          0
178 001AC D             NIBHEX D
179               *
180 001AD 2B0           CON(3)  178             21  ENABLE INTR <interrupt mask by
```

```
181 001B0 0000          REL(5) =ENABLE
          0
182 001B5 D             NIBHEX D
183               *
184 001B6 BE1           CON(3)  491             22  STANDBY [ON | OFF] or value
185 001B9 0000          REL(5) =STANBY
          0
186 001BE D             NIBHEX D
187               *
188               =tCNTRL EQU    #23
189 001BF 160           CON(3)   97             23  CONTROL ON|OFF
190 001C2 0000          REL(5) =CONTRL
          0
191 001C7 D             NIBHEX D
192               *
193               =tIO    EQU    #24
194 001C8 AE0           CON(3)  234             24  (See OFF, ASSIGN, and RESTORE)
195 001CB 0000          NIBHEX 00000
          0
196 001D0 0             NIBHEX 0
197               *
198               =tLOCK0 EQU    #25
199 001D1 901           CON(3)  265             25  (See LOCAL)
200 001D4 0000          NIBHEX 00000
          0
201 001D9 0             NIBHEX 0
202               *
203               =tINTRR EQU    #26
204 001DA FD0           CON(3)  223             26  (See ON/OFF)
205 001DD 0000          NIBHEX 00000
          0
206 001E2 0             NIBHEX 0
```

```
    207                     STITLE T e x t  T a b l e
    208              *  Text Table
    209 001E3         TxTbSt                        Text table start
    210              *
    211 001E3 B             NIBHEX B                ASSIGN IO
    212 001E4 1435          NIBASC \ASSIGN\
              3594
              74E4
    213 001F0 F0            NIBHEX F0
    214              *
    215 001F2 B             NIBHEX B                BINAND ( <int.exp> ,
    216 001F3 2494          NIBASC \BINAND\
              E414
              E444
    217 001FF 10            NIBHEX 10
    218              *
    219 00201 B             NIBHEX B                BINCMP ( <int.exp> )
    220 00202 2494          NIBASC \BINCMP\
              E434
              D405
    221 0020E 20            NIBHEX 20
    222              *
    223 00210 B             NIBHEX B                BINEOR ( <int.exp> ,
    224 00211 2494          NIBASC \BINEOR\
              E454
              F425
    225 0021D 30            NIBHEX 30
    226              *
    227 0021F B             NIBHEX B                BINIOR ( <int.exp> ,
    228 00220 2494          NIBASC \BINIOR\
              E494
              F425
    229 0022C 40            NIBHEX 40
    230              *
    231 0022E 5             NIBHEX 5                BIT ( <int.exp> , <b
    232 0022F 2494          NIBASC \BIT\
              45
    233 00235 50            NIBHEX 50
    234              *
    235 00237 9             NIBHEX 9                CLEAR LOOP[;<loop>]
    236 00238 34C4          NIBASC \CLEAR\
              5414
              25
    237 00242 E0            NIBHEX E0
    238              *
    239 00244 D             NIBHEX D                CONTROL ON|OFF
    240 00245 34F4          NIBASC \CONTROL\
              E445
              25F4
              C4
    241 00253 32            NIBHEX 32
    242              *
    243 00255 D             NIBHEX D                A=DEVADDR(<device sp
    244 00256 4454          NIBASC \DEVADDR\
              6514
```

```
            4444
            25
    245 00264 60          NIBHEX 60
    246           *
    247 00266 B           NIBHEX B              A=DEVAID(<device spe
    248 00267 4454        NIBASC \DEVAID\
            6514
            9444
    249 00273 80          NIBHEX 80
    250           *
    251 00275 B           NIBHEX B              A$=DEVID$(<device sp
    252 00276 4454        NIBASC \DEVID$\
            6594
            4442
    253 00282 70          NIBHEX 70
    254           *
    255 00284 D           NIBHEX D              DISPLAY IS code
    256 00285 4494        NIBASC \DISPLAY\
            3505
            C414
            95
    257 00293 91          NIBHEX 91
    258           *
    259 00295 B           NIBHEX B              ENABLE INTR <interru
    260 00296 54E4        NIBASC \ENABLE\
            1424
            C454
    261 002A2 12          NIBHEX 12
    262           *
    263 002A4 9           NIBHEX 9              ENTER <dev spec> [US
    264 002A5 54E4        NIBASC \ENTER\
            4554
            25
    265 002AF 41          NIBHEX 41
    266           *
    267 002B1 D           NIBHEX D              INITIALIZE [<volume>
    268 002B2 94E4        NIBASC \INITIAL\
            9445
            9414
            C4
    269 002C0 D0          NIBHEX D0
    270           *
    271 002C2 7           NIBHEX 7              (See ON/OFF)
    272 002C3 94E4        NIBASC \INTR\
            4525
    273 002CB 62          NIBHEX 62
    274           *
    275 002CD 3           NIBHEX 3              (See OFF, ASSIGN, an
    276 002CE 94F4        NIBASC \IO\
    277 002D2 42          NIBHEX 42
    278           *
    279 002D4 7           NIBHEX 7              LIST IO
    280 002D5 C494        NIBASC \LIST\
            3545
    281 002DD 21          NIBHEX 21
```

```
282                *
283 002DF 9            NIBHEX 9              LOCAL [<dev.spec|loo
284 002E0 C4F4         NIBASC \LOCAL\
          3414
          C4
285 002EA D1           NIBHEX D1
286                *
287 002EC D            NIBHEX D              (See LOCAL)
288 002ED C4F4         NIBASC \LOCKOUT\
          34B4
          F455
          45
289 002FB 52           NIBHEX 52
290                *
291 002FD 5            NIBHEX 5              OFF IO
292 002FE F464         NIBASC \OFF\
          64
293 00304 01           NIBHEX 01
294                *
295 00306 3            NIBHEX 3              ON INTR GOSUB/GOTO <
296 00307 F4E4         NIBASC \ON\
297 0030B 51           NIBHEX 51
298                *
299 0030D B            NIBHEX B              OUTPUT <dev spec> [U
300 0030E F455         NIBASC \OUTPUT\
          4505
          5545
301 0031A 31           NIBHEX 31
302                *
303 0031C D            NIBHEX D              PACKDIR <Device spec
304 0031D 0514         NIBASC \PACKDIR\
          34B4
          4494
          25
305 0032B B1           NIBHEX B1
306                *
307 0032D 7            NIBHEX 7              PACK <Device specifi
308 0032E 0514         NIBASC \PACK\
          34B4
309 00336 A1           NIBHEX A1
310                *
311 00338 7            NIBHEX 7              PASS CONTROL <dev. s
312 00339 0514         NIBASC \PASS\
          3535
313 00341 02           NIBHEX 02
314                *
315 00343 D            NIBHEX D              PRINTER IS code
316 00344 0525         NIBASC \PRINTER\
          94E4
          4554
          25
317 00352 81           NIBHEX 81
318                *
319 00354 D            NIBHEX D              READDDC {read last D
320 00355 2554         NIBASC \READDDC\
```

```
                1444
                4444
                34
    321 00363 B0          NIBHEX B0
    322             *
    323 00365 F           NIBHEX F                READINTR {read inter
    324 00366 2554        NIBASC \READINTR\
                1444
                94E4
                4525
    325 00376 A0          NIBHEX A0
    326             *
    327 00378 B           NIBHEX B                REMOTE [<dev.spec>..
    328 00379 2554        NIBASC \REMOTE\
                D4F4
                4554
    329 00385 E1          NIBHEX E1
    330             *
    331 00387 D           NIBHEX D                REQUEST [<loop #>;]<
    332 00388 2554        NIBASC \REQUEST\
                1555
                5435
                45
    333 00396 C1          NIBHEX C1
    334             *
    335 00398 9           NIBHEX 9                RESET HPIL
    336 00399 2554        NIBASC \RESET\
                3554
                45
    337 003A3 71          NIBHEX 71
    338             *
    339 003A5 D           NIBHEX D                RESTORE IO
    340 003A6 2554        NIBASC \RESTORE\
                3545
                F425
                54
    341 003B4 11          NIBHEX 11
    342             *
    343 003B6 7           NIBHEX 7                SEND [;<loop>] {<Fra
    344 003B7 3554        NIBASC \SEND\
                E444
    345 003BF 61          NIBHEX 61
    346             *
    347 003C1 9           NIBHEX 9                SPOLL ( <device spec
    348 003C2 3505        NIBASC \SPOLL\
                F4C4
                C4
    349 003CC 90          NIBHEX 90
    350             *
    351 003CE D           NIBHEX D                STANDBY [ON | OFF] o
    352 003CF 3545        NIBASC \STANDBY\
                14E4
                4424
                95
    353 003DD 22          NIBHEX 22
```

```
354                *
355 003DF B            NIBHEX B              STATUS [( <loop #> )
356 003E0 3545         NIBASC \STATUS\
          1445
          5535
357 003EC C0           NIBHEX C0
358                *
359 003EE D            NIBHEX D              TRIGGER [<dev.spec>.
360 003EF 4525         NIBASC \TRIGGER\
          9474
          7454
          25
361 003FD F1           NIBHEX F1
362 003FF 1FF   TxTbEn NIBHEX 1FF           Text termination
363 00402              END
```

ASGNIO	Ext		-	109							
BINAND	Ext		-	53							
BINCMP	Ext		-	57							
BINEOR	Ext		-	61							
BINIOR	Ext		-	65							
BIT	Ext		-	69							
CLEAR	Ext		-	105							
CONTRL	Ext		-	190							
ChainE	Ext		-	9							
DEVID	Ext		-	77							
DEVTYP	Ext		-	81							
DISPIS	Ext		-	149							
ENABLE	Ext		-	181							
ENTER	Ext		-	129							
FIND	Ext		-	73							
INITXQ	Ext		-	101							
LISTIO	Ext		-	121							
LOCAL	Ext		-	165							
OFFIO	Ext		-	113							
ONINTx	Ext		-	133							
OUTPUT	Ext		-	125							
PACK	Ext		-	153							
PACKD	Ext		-	157							
PASS	Ext		-	177							
PILMSG	Ext		-	46							
PILPOL	Ext		-	47							
PRNTIS	Ext		-	145							
READDC	Ext		-	93							
READIN	Ext		-	89							
REMOTE	Ext		-	169							
REQST	Ext		-	161							
RESET	Ext		-	141							
RESTIO	Ext		-	117							
SEND	Ext		-	137							
SPOLL	Ext		-	85							
STANBY	Ext		-	185							
STATUS	Ext		-	97							
TRIGER	Ext		-	173							
TxTbEn	Rel	1023 #003FF	-	362	23	24	25	27	28	30	31
				34	38	39	40	41	42	43	
TxTbSt	Rel	483 #001E3	-	209	23	24	25	27	28	30	31
				34	38	39	40	41	42	43	45
fLEX	Ext		-	5							
=tCNTRL	Abs	35 #00023	-	188							
=tINTRR	Abs	38 #00026	-	203							
=tIO	Abs	36 #00024	-	193							
=tLOCKO	Abs	37 #00025	-	198							
=xromFF	Rel	141 #0008D	-	50							

Input Parameters

  Source file name is NZ&TBL::MS

  Listing file name is NZ/TBL:TI:ML

  Object file name is NZ%TBL:TI:MS

```
                                    111111
                          0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1                *
 2                *       N   N  ZZZZZ   &     EEEEE  RRRR   RRRR
 3                *       N   N      Z  & &    E      R   R  R   R
 4                *       NN  N     Z   & &    E      R   R  R   R
 5                *       N N N    Z     &     EEEE   RRRR   RRRR
 6                *       N  NN   Z     & & &  E      R R    R R
 7                *       N   N  Z      &  &   E      R  R   R  R
 8                *       N   N  ZZZZZ   && &  EEEEE  R   R  R   R
 9                *
10                *
11                * Date of last update <830929.1738>
12                *
13                * HPIL uses error numbers in the range 0-63 (0-3F Hex)
14                * (Error numbers between 64 and (end) are building blocks)
15                *
16 00000 10              CON(2)   1       Min message #
17 00002 34              CON(2)  67       Max message #
18                *
19                =eHPIL  EQU    00              (TITLE for my errors)
20 00004 01              CON(2)  16
21 00006 00              CON(2) 00               Message number 00
22 00008 4               CON(1)   4
23 00009 8405            NIBASC \HPIL \
         94C4
         02
24 00013 C               CON(1)  12
25                *
26                * Errors 1-15 are parse errors
27                *
28                *
29                =eNOASN EQU    01              ASSIGN IO Needed
30 00014 51              CON(2)  21
31 00016 10              CON(2) 01               Message number 01
32 00018 5               CON(1)   5
33 00019 1435            NIBASC \ASSIGN\
         3594
         74E4
34 00025 D               CON(1)  13
35 00026 34              CON(2) =eION
36 00028 C               CON(1)  12
37                *
38                =eXCESS EQU    03              Excess chars
39 00029 80              CON(2)   8
40 0002B 30              CON(2) 03               Message number 03
41 0002D E               CON(1)  14
42 0002E 00              CON(2) =eEXCHR
43 00030 C               CON(1)  12
44                *
45                =eMSPAr EQU    04              Missing parameter(s)
46 00031 80              CON(2)   8
47 00033 40              CON(2) 04               Message number 04
48 00035 E               CON(1)  14
49 00036 00              CON(2) =eMSPAR
50 00038 C               CON(1)  12
51                *
```

```
 52                =eILPAr EQU    05             Illegal parameter(s)
 53 00039 80               CON(2)   8
 54 0003B 50               CON(2) 05              Message number 05
 55 0003D E                CON(1)  14
 56 0003E 00 .             CON(2) =eILPAR
 57 00040 C                CON(1)  12
 58                *
 59                =eILEXp EQU    06             Illegal expression
 60 00041 80               CON(2)   8
 61 00043 60               CON(2) 06              Message number 06
 62 00045 E                CON(1)  14
 63 00046 00               CON(2) =eILEXP
 64 00048 C                CON(1)  12
 65                *
 66                =eSYNTx EQU    07             Syntax Error
 67 00049 80               CON(2)   8
 68 0004B 70               CON(2) 07              Message number 07
 69 0004D E                CON(1)  14
 70 0004E 00               CON(2) =eSYNTX
 71 00050 C                CON(1)  12
 72                *
 73                * Errors 8-15 are reserved
 74                *
 75                * Errors 16-31 are tape errors
 76                *
 77                *
 78                =efPROT EQU    16             File Protect
 79 00051 80               CON(2)   8
 80 00053 01               CON(2) 16              Message number 16
 81 00055 E                CON(1)  14
 82 00056 00               CON(2) =eFPROT
 83 00058 C                CON(1)  12
 84                *
 85                =eEOTAP EQU    17             End of medium
 86 00059 71               CON(2)  23
 87 0005B 11               CON(2) 17              Message number 17
 88 0005D 6                CON(1)   6
 89 0005E 54E6             NIBASC \End Of \
          4602
          F466
          02
 90 0006C D                CON(1)  13
 91 0006D 24               CON(2) =eMEDIA
 92 0006F C                CON(1)  12
 93                =eINVAL EQU    18            Invalid medium
 94                *
 95                =eSTALL EQU    18             Tape stall-Invalid medium
 96 00070 B0               CON(2)  11
 97 00072 21               CON(2) 18              Message number 18
 98 00074 E                CON(1)  14
 99 00075 00               CON(2) =eINVLD
100 00077 D                CON(1)  13
101 00078 24               CON(2) =eMEDIA
102 0007A C                CON(1)  12
103                *
```

```
104                =eNOLIF EQU    19          Not LIF-Invalid medium
105 0007B 80               CON(2)   8
106 0007D 31               CON(2) 19          Message number 19
107 0007F D                CON(1)  13
108 00080 21               CON(2) =eINVAL
109 00082 C                CON(1)  12
110                *
111                =eNOTAP EQU    20          No medium
112 00083 F0               CON(2)  15
113 00085 41               CON(2) 20          Message number 20
114 00087 2                CON(1)   2
115 00088 E4F6             NIBASC \No \
          02
116 0008E D                CON(1)  13
117 0008F 24               CON(2) =eMEDIA
118 00091 C                CON(1)  12
119                *
120                =eNFILE EQU    22          File not found
121 00092 80               CON(2)   8
122 00094 61               CON(2) 22          Message number 22
123 00096 E                CON(1)  14
124 00097 00               CON(2) =eFnFND
125 00099 C                CON(1)  12
126                *
127                =eNEWTA EQU    23          New medium-Invalid medium
128 0009A 80               CON(2)   8
129 0009C 71               CON(2) 23          Message number 23
130 0009E D                CON(1)  13
131 0009F 21               CON(2) =eINVAL
132 000A1 C                CON(1)  12
133                *
134                =eBLANK EQU    24          No data -Invalid medium
135 000A2 80               CON(2)   8
136 000A4 81               CON(2) 24          Message number 24
137 000A6 D                CON(1)  13
138 000A7 21               CON(2) =eINVAL
139 000A9 C                CON(1)  12
140                *
141                =eRECRD EQU    25          Record #-Invalid medium
142 000AA 80               CON(2)   8
143 000AC 91               CON(2) 25          Message number 25
144 000AE D                CON(1)  13
145 000AF 21               CON(2) =eINVAL
146 000B1 C                CON(1)  12
147                *
148                =eCHSUM EQU    26          Checksum-Invalid medium
149 000B2 80               CON(2)   8
150 000B4 A1               CON(2) 26          Message number 26
151 000B6 D                CON(1)  13
152 000B7 21               CON(2) =eINVAL
153 000B9 C                CON(1)  12
154                *
155                =eTSIZE EQU    28          Size of file
156 000BA 91               CON(2)  25
157 000BC C1               CON(2) 28          Message number 28
```

```
    158 000BE 7           CON(1)   7
    159 000BF 3596        NIBASC \Size of \
              A756
              02F6
              6602
    160 000CF E           CON(1)  14
    161 000D0 00          CON(2) =eFILE
    162 000D2 C           CON(1)  12
    163             *
    164             =eEFILE EQU    30             File exists
    165 000D3 80          CON(2)   8
    166 000D5 E1          CON(2) 30              Message number 30
    167 000D7 E           CON(1)  14
    168 000D8 00          CON(2) =eFEXST
    169 000DA C           CON(1)  12
    170             *
    171             =eDIRFL EQU    31             Directory full
    172 000DB 32          CON(2)  35
    173 000DD F1          CON(2) 31              Message number 31
    174 000DF B           CON(1)  11
    175 000E0 D           CON(1)  13
    176 000E1 4496        NIBASC \Director\
              2756
              3647
              F627
    177 000F1 9702        NIBASC \y Full\
              6457
              C6C6
    178 000FD C           CON(1)  12
    179             *
    180             * Errors 32-47 are HPIL Errors
    181             *
    182             =eNOFND EQU    32            Device not found
    183             *
    184             =eTERM  EQU    32             (Terminator match)
    185 000FE 80          CON(2)   8
    186 00100 02          CON(2) 32              Message number 32
    187 00102 E           CON(1)  14
    188 00103 00          CON(2) =eDVCNF
    189 00105 C           CON(1)  12
    190             *
    191             =eNORDY EQU    34             Device not ready
    192 00106 B1          CON(2)  27
    193 00108 22          CON(2) 34              Message number 34
    194 0010A D           CON(1)  13
    195 0010B 14          CON(2) =eDEVIC
    196 0010D 8           CON(1)   8
    197 0010E E4F6        NIBASC \Not Read\
              4702
              2556
              1646
    198 0011E 97          NIBASC \y\
    199 00120 C           CON(1)  12
    200             *
    201             =eLTIMO EQU    35             Loop broken
```

```
202 00121 C1          CON(2)  28
203 00123 32          CON(2) 35           Message number 35
204 00125 A           CON(1)  10
205 00126 C4F6        NIBASC \Loop Bro\
          F607
          0224
          27F6
206 00136 B656        NIBASC \ken\
          E6
207 0013C C           CON(1)  12
208             *
209             =eFLOST EQU    36         Frame Error
210 0013D 31          CON(2)  19
211 0013F 42          CON(2) 36           Message number 36
212 00141 D           CON(1)  13
213 00142 04          CON(2) =eFRAME
214 00144 4           CON(1)   4
215 00145 5427        NIBASC \Error\
          27F6
          27
216 0014F C           CON(1)  12
217             *
218             =eOVRUN EQU    37         Frame Overrun
219 00150 80          CON(2)   8
220 00152 52          CON(2) 37           Message number 37
221 00154 D           CON(1)  13
222 00155 42          CON(2) =eFLOST
223 00157 C           CON(1)  12
224             *
225             =eLPERR EQU    38         Frame Changed
226 00158 80          CON(2)   8
227 0015A 62          CON(2) 38           Message number 38
228 0015C D           CON(1)  13
229 0015D 42          CON(2) =eFLOST
230 0015F C           CON(1)  12
231             *
232             =eUNEXP EQU    39         Unexpected Frame
233 00160 F1          CON(2)  31
234 00162 72          CON(2) 39           Message number 39
235 00164 A           CON(1)  10
236 00165 55E6        NIBASC \Unexpect\
          5687
          0756
          3647
237 00175 5646        NIBASC \ed \
          02
238 0017B D           CON(1)  13
239 0017C 04          CON(2) =eFRAME
240 0017E C           CON(1)  12
241             *
242             =eXXXXX EQU    40         Frame Lost
243 0017F 80          CON(2)   8
244 00181 82          CON(2) 40           Message number 40
245 00183 D           CON(1)  13
246 00184 42          CON(2) =eFLOST
```

```
247 00186 C           CON(1)  12
248           *
249           =eBADMD EQU   41          Invalid Mode
250 00187 11          CON(2)  17
251 00189 92          CON(2) 41         Message number 41
252 0018B E           CON(1)  14
253 0018C 00          CON(2) =eINVLD
254 0018E 3           CON(1)   3
255 0018F D4F6        NIBASC \Mode\
          4656
256 00197 C           CON(1)  12
257           *
258           =eFRTOI EQU   42          Frame Timeout (SCI)
259 00198 80          CON(2)   8
260 0019A A2          CON(2) 42         Message number 42
261 0019C D           CON(1)  13
262 0019D 32          CON(2) =eLTIMO
263 0019F C           CON(1)  12
264           *
265           =eFRTOL EQU   43          Frame Timeout (Loop)
266 001A0 80          CON(2)   8
267 001A2 B2          CON(2) 43         Message number 43
268 001A4 D           CON(1)  13
269 001A5 32          CON(2) =eLTIMO
270 001A7 C           CON(1)  12
271           *
272           =eSYSer EQU   44          System Error (Bad cur addr)
273 001A8 80          CON(2)   8
274 001AA C2          CON(2) 44         Message number 44
275 001AC E           CON(1)  14
276 001AD 00          CON(2) =eMMCOR
277 001AF C           CON(1)  12
278           *
279           =eTESTF EQU   45          Selftest failed
280 001B0 72          CON(2)  39
281 001B2 D2          CON(2) 45         Message number 45
282 001B4 B           CON(1)  11
283 001B5 F           CON(1)  15
284 001B6 3556        NIBASC \Self-tes\
          C666
          D247
          5637
285 001C6 4702        NIBASC \t failed\
          6616
          96C6
          5646
286 001D6 C           CON(1)  12
287           *
288           =eDTYPE EQU   47          Device type
289 001D7 11          CON(2)  17
290 001D9 F2          CON(2) 47         Message number 47
291 001DB D           CON(1)  13
292 001DC 14          CON(2) =eDEVIC
293 001DE 3           CON(1)   3
294 001DF 4597        NIBASC \Type\
```

```
          0756
295 001E7 C           CON(1)  12
296               *
297               * Errors 48-50 are unused
298               *
299               *
300               * Error 51 is reserved
301               *
302               *
303               =eABORT EQU    52             Aborted operation
304 001E8 41          CON(2)  20
305 001EA 43          CON(2) 52              Message number 52
306 001EC 6           CON(1)   6
307 001ED 1426        NIBASC \Aborted\
          F627
          4756
          46
308 001FB C           CON(1)  12
309               *
310               =eDSPEC EQU    53             Invalid device spec
311 001FC 41          CON(2)  20
312 001FE 53          CON(2) 53              Message number 53
313 00200 E           CON(1)  14
314 00201 00          CON(2) =eINVLD
315 00203 D           CON(1)  13
316 00204 14          CON(2) =eDEVIC
317 00206 3           CON(1)   3
318 00207 3507        NIBASC \Spec\
          5636
319 0020F C           CON(1)  12
320               *
321               =eNNUMR EQU    54             Not numeric
322 00210 80          CON(2)   8
323 00212 63          CON(2) 54              Message number 54
324 00214 E           CON(1)  14
325 00215 00          CON(2) =eDATTY
326 00217 C           CON(1)  12
327               *
328               =eRANGE EQU    56             Invalid Arg
329 00218 80          CON(2)   8
330 0021A 83          CON(2) 56              Message number 56
331 0021C E           CON(1)  14
332 0021D 00          CON(2) =eIVARG
333 0021F C           CON(1)  12
334               *
335               =eNMBOX EQU    57             No loop
336 00220 41          CON(2)  20
337 00222 93          CON(2) 57              Message number 57
338 00224 6           CON(1)   6
339 00225 E4F6        NIBASC \No Loop\
          02C4
          F6F6
          07
340 00233 C           CON(1)  12
341               *
```

```
342                =eNORAM EQU    59              Insufficient memory
343 00234 80               CON(2)   8
344 00236 B3               CON(2) 59              Message number 59
345 00238 E                CON(1)  14
346 00239 00               CON(2) =eMEM
347 0023B C                CON(1)  12
348                *
349                =eOFFED EQU    60              RESTORE IO Needed
350 0023C 71               CON(2)  23
351 0023E C3               CON(2) 60              Message number 60
352 00240 6                CON(1)   6
353 00241 2554             NIBASC \RESTORE\
          3545
          F425
          54
354 0024F D                CON(1)  13
355 00250 34               CON(2) =eION
356 00252 C                CON(1)  12
357                *
358                * Errors 61-63 are reserved
359                *
360                * Error messages 64-end are building blocks
361                *
362                *
363                =eFRAME EQU    64              "Message" Building block
364 00253 61               CON(2)  22
365 00255 04               CON(2) 64              Message number 64
366 00257 7                CON(1)   7
367 00258 D456             NIBASC \Message \
          3737
          1676
          5602
368 00268 C                CON(1)  12
369                *
370                =eDEVIC EQU    65              "Device " building block
371 00269 41               CON(2)  20
372 0026B 14               CON(2) 65              Message number 65
373 0026D 6                CON(1)   6
374 0026E 4456             NIBASC \Device \
          6796
          3656
          02
375 0027C C                CON(1)  12
376                *
377                =eMEDIA EQU    66              "Medium" building block
378 0027D 21               CON(2)  18
379 0027F 24               CON(2) 66              Message number 66
380 00281 5                CON(1)   5
381 00282 D456             NIBASC \Medium\
          4696
          5706
382 0028E C                CON(1)  12
383                *
384                =eION   EQU    67              " IO Needed" building block
385 0028F A1               CON(2)  26
```

```
    386 00291 34          CON(2) 67           Message number 67
    387 00293 9           CON(1)   9
    388 00294 0294        NIBASC \ IO Need\
              F402
              E456
              5646
    389 002A4 5646        NIBASC \ed\
    390 002A8 C           CON(1)  12
    391             *
    392 002A9 FF          NIBHEX FF           Table terminator
    393 002AB             END
```

```
=eABORT  Abs      52 #00034 -   303
=eBADMD  Abs      41 #00029 -   249
=eBLANK  Abs      24 #00018 -   134
=eCHSUM  Abs      26 #0001A -   148
 eDATTY  Ext                -   325
=eDEVIC  Abs      65 #00041 -   370   195   292   316
=eDIRFL  Abs      31 #0001F -   171
=eDSPEC  Abs      53 #00035 -   310
=eDTYPE  Abs      47 #0002F -   288
 eDVCNF  Ext                -   188
=eEFILE  Abs      30 #0001E -   164
=eEOTAP  Abs      17 #00011 -    85
 eEXCHR  Ext                -    42
 eFEXST  Ext                -   168
 eFILE   Ext                -   161
=eFLOST  Abs      36 #00024 -   209   222   229   246
 eFPROT  Ext                -    82
=eFRAME  Abs      64 #00040 -   363   213   239
=eFRTOI  Abs      42 #0002A -   258
=eFRTOL  Abs      43 #0002B -   265
 eFnFND  Ext                -   124
=eHPIL   Abs       0 #00000 -    19
 eILEXP  Ext                -    63
=eILEXp  Abs       6 #00006 -    59
 eILPAR  Ext                -    56
=eILPAr  Abs       5 #00005 -    52
=eINVAL  Abs      18 #00012 -    93   108   131   138   145   152
 eINVLD  Ext                -    99   253   314
=eION    Abs      67 #00043 -   384    35   355
 eIVARG  Ext                -   332
=eLPERR  Abs      38 #00026 -   225
=eLTIMO  Abs      35 #00023 -   201   262   269
=eMEDIA  Abs      66 #00042 -   377    91   101   117
 eMEM    Ext                -   346
 eMMCOR  Ext                -   276
 eMSPAR  Ext                -    49
=eMSPAr  Abs       4 #00004 -    45
=eNEWTA  Abs      23 #00017 -   127
=eNFILE  Abs      22 #00016 -   120
=eNMBOX  Abs      57 #00039 -   335
=eNNUMR  Abs      54 #00036 -   321
=eNOASN  Abs       1 #00001 -    29
=eNOFND  Abs      32 #00020 -   182
=eNOLIF  Abs      19 #00013 -   104
=eNORAM  Abs      59 #0003B -   342
=eNORDY  Abs      34 #00022 -   191
=eNOTAP  Abs      20 #00014 -   111
=eOFFED  Abs      60 #0003C -   349
=eOVRUN  Abs      37 #00025 -   218
=eRANGE  Abs      56 #00038 -   328
=eRECRD  Abs      25 #00019 -   141
=eSTALL  Abs      18 #00012 -    95
 eSYNTX  Ext                -    70
=eSYNTx  Abs       7 #00007 -    66
=eSYSer  Abs      44 #0002C -   272
```

```
=eTERM   Abs      32 #00020 -   184
=eTESTF  Abs      45 #0002D -   279
=eTSIZE  Abs      28 #0001C -   155
=eUNEXP  Abs      39 #00027 -   232
=eXCESS  Abs       3 #00003 -    38
=eXXXXX  Abs      40 #00028 -   242
=efPROT  Abs      16 #00010 -    78
```

Input Parameters

Source file name is NZ&ERR::MS

Listing file name is NZ/ERR:TI:ML::-1

Object file name is NZ%ERR:TI:MS::-1

```
                                    111111
                          0123456789012345
```

Initial flag settings are

Errors

None

Saturn Assembler News

```
                *
                *       N   N  ZZZZZ   &     DDDD   III  RRRR
                *       N   N      Z  & &     D  D   I   R   R
                *       NN  N     Z   & &     D  D   I   R   R
                *       N N N    Z     &      D  D   I   RRRR
                *       N  NN   Z     & & &   D  D   I   R R
                *       N   N  Z      &  &    D  D   I   R  R
                *       N   N  ZZZZZ   && &  DDDD   III  R   R
                *
                *
                        TITLE  DIRECTORY SECTION <840106.1804>
F06B5           ABS    #F06B5        TI%HP6 address (fixed)
                ****************************************************************
                ****************************************************************
                **
                ** Name:       PILPOL - Poll handler for HPIL ROM (calls others)
                **
                ** Category:   PILUTL
                **
                ** Purpose:
                **      Handle the POLL entry (check if this is a poll I
                **      respond to...if so, jump to the poll handler for that
                **      specific poll
                **
                ** Entry:
                **      B[A] is the poll number
                **
                ** Exit:
                **      If not handled:
                **        XM=1, carry clear
                **      If handled successfully:
                **        XM=0, carry clear
                **      If error during handling:
                **        Carry set
                **
                ** Calls:      None
                **
                ** Uses.......
                **  Inclusive: B[A],C[A]
                **
                ** Stk lvls:   1 (internal GOSUB){Specific handlers may be more}
                **
                ** History:
                **
                **    Date     Programmer              Modification
                **   --------   ----------    ---------------------------------
                **  09/26/83      NZ         Added documentation
                **
                ****************************************************************
                ****************************************************************
F06B5 20        =PILPOL P=     0
F06B7 D2                C=0    A
F06B9 31E1              LC(2)  ((TEND)-(TSTART))/5  Number of table entries
F06BD 04                SETHEX                Just to be SURE
F06BF 8B5               ?B<C   A
```

```
     56 F06C2 60          GOYES  POLCH1       Check if ROM entry point
     57 F06C4 6EB0        GOTO   POLCHR       ROM entry point
     58               *
     59               * Now compute the offset to the Poll handler
     60               *
     61 F06C8 7690 POLCH1 GOSUB  POLCH2       Set RSTK=TSTART (to get address)
     62               *
     63               * This is the jump table
     64               *
     65 F06CC      TSTART
     66 F06CC 0000        REL(5) =hVER$       #00 VER$
              0
     67 F06D1 0000        REL(5) =DEVSPp      #01 Device parse
              0
     68 F06D6 0000        REL(5) =PILDC       #02 File decompile
              0
     69 F06DB 4B00        REL(5) =RTNSXM      #03 Device execute
              0
     70 F06E0 0000        REL(5) =FILSPp      #04 File spec parse
              0
     71 F06E5 0000        REL(5) =FILSPx      #05 File spec XEQ
              0
     72 F06EA 0000        REL(5) =hCAT        #06 CAT
              0
     73 F06EF 0000        REL(5) =hCAT$       #07 CAT$
              0
     74 F06F4 0000        REL(5) =hCOPYx      #08 COPY execute
              0
     75 F06F9 0000        REL(5) =hCREAT      #09 Create XEQ
              0
     76 F06FE 0000        REL(5) =hDIDST      #0A Device ID store (HPIL)
              0
     77 F0703 0000        REL(5) =hFPROT      #0B Private/Secure/Unsecure
              0
     78 F0708 7800        REL(5) =RTNSXM      #0C LIST (File not in mainframe)
              0
     79 F070D 2800        REL(5) =RTNSXM      #0D MERGE (File not in mainframe)
              0
     80 F0712 0000        REL(5) =hPRTCL      #0E Print class
              0
     81 F0717 0000        REL(5) =PRTIS       #0F Print (part 1)
              0
     82 F071C 0000        REL(5) =hPURGE      #10 PURGE
              0
     83 F0721 0000        REL(5) =hRENAM      #11 ReNAME
              0
     84 F0726 0000        REL(5) =hENTER      #12 Enter
              0
     85 F072B 4600        REL(5) =RTNSXM      #13 HPIL poll 2
              0
     86 F0730 F500        REL(5) =RTNSXM      #14 HPIL poll 3
              0
     87 F0735 A500        REL(5) =RTNSXM      #15 HPIL poll 4
              0
     88 F073A 5500        REL(5) =RTNSXM      #16 HPIL poll 5
```

```
          0
     89 F073F 0000        REL(5) =hFINDF       #17 Find file
          0
     90 F0744 0000        REL(5) =hRDCBF       #18 Read current record to file bufr
          0
     91 F0749 0000        REL(5) =hRDNBF       #19 Write bufr out & read next recor
          0
     92 F074E 0000        REL(5) =hWRCBF       #1A Write file bufr to current recor
          0
     93 F0753 0000        REL(5) =hKYDF        #1B Build key defn
          0
     94 F0758 7300        REL(5) =RTNSXM       #1C WTKY - waiting for key in KEYRD
          0
     95 F075D 0000        REL(5) =ENTUSG       #1D IMAGE execution starts
          0
     96                   *
     97                   * End of polls handled by HPIL ROM
     98                   *
     99 F0762             TEND
    100                   *
    101                   * REMAINING CODE FOR TABLE LOOKUP
    102                   *
    103 F0762 07   POLCH2  C=RSTK
    104 F0764 C9           C=B+C  A
    105 F0766 C5           B=B+B  A
    106 F0768 C5           B=B+B  A             B*4
    107 F076A C9           C=B+C  A             C[A] is now address of jump address
    108                   *
    109 F076C D5           B=C    A             Save address in B[A] for offset
    110 F076E 137          CD1EX                D1 @ address, D1 value in C[A]
    111 F0771 06           RSTK=C               Push D1 value (to allow restore)
    112 F0773 147          C=DAT1 A             Read offset to actual address
    113 F0776 C1           B=C+B  A             B[A] is address of specific handler
    114 F0778 07           C=RSTK               Restore D1 from RSTK...
    115 F077A 135          D1=C                 ...to D1
    116 F077D D9           C=B    A             Copy address to C[A]...
    117 F077F 06           RSTK=C               ...Push address onto stack...
    118 F0781 03           RTNCC                ...and jump to the routine
    119                   *-
    120                   *-
    121                   *
    122                   * Check for system polls (#F0 through #FF)
    123                   *
    124 F0783 BED  POLCHR  B=-B-1 B             Ones complement of poll # in B[A]
    125 F0786 3190         LC(2)  ((TEND2)-(TSTAR2))/5  Load # of ROM entries
    126 F078A 8B5          ?B<C   A             In the range HPIL knows?
    127 F078D 40           GOYES  POLCH3        Yes...compute specific handler addr
    128 F078F 00   RTNSXM  RTNSXM               No...return, carry clear, XM=1
    129                   *-
    130                   *-
    131 F0791 7DCF POLCH3  GOSUB  POLCH2        Same driver, given the table addr
    132                   *
    133                   * This is the table for system polls
    134                   *
    135 F0795             TSTAR2
```

```
136 F0795 0000          REL(5) =PILCST       #FF CLDST Cold start address
          0
137 F079A 0000          REL(5) =PILWNK       #FE DSWNK Deep sleep wakeup-no key
          0
138 F079F 0000          REL(5) =PILWKP       #FD DSWKY Deep sleep wakeup
          0
139 F07A4 0000          REL(5) =PILPOF       #FC PWROF Power off
          0
140 F07A9 0000          REL(5) =PILCNF       #FB CONFG Configuration
          0
141 F07AE 0000          REL(5) =PILMLP       #FA MNLP Main loop
          0
142 F07B3 0000          REL(5) =PILSRQ       #F9 SREQ Service request
          0
143 F07B8 0000          REL(5) =hEXCPT       #F8 Excpt Exception check after stmt
          0
144 F07BD 0000          REL(5) =hZERPG       #F7 ZERPG The Math stack is collapse
          0
145               *
146               * End of polls handled by HPIL ROM
147               *
148 F07C2         TEND2
149 F07C2                 END
```

DEVSPp	Ext			-	67							
ENTUSG	Ext			-	95							
FILSPp	Ext			-	70							
FILSPx	Ext			-	71							
PILCNF	Ext			-	140							
PILCST	Ext			-	136							
PILDC	Ext			-	68							
PILMLP	Ext			-	141							
PILPOF	Ext			-	139							
=PILPOL	Abs	984757	#F06B5	-	51							
PILSRQ	Ext			-	142							
PILWKP	Ext			-	138							
PILWNK	Ext			-	137							
POLCH1	Abs	984776	#F06C8	-	61	56						
POLCH2	Abs	984930	#F0762	-	103	61	131					
POLCH3	Abs	984977	#F0791	-	131	127						
POLCHR	Abs	984963	#F0783	-	124	57						
PRTIS	Ext			-	81							
RTNSXM	Abs	984975	#F078F	-	128	69	78	79	85	86	87	88
					94							
TEND	Abs	984930	#F0762	-	99	53						
TEND2	Abs	985026	#F07C2	-	148	125						
TSTAR2	Abs	984981	#F0795	-	135	125						
TSTART	Abs	984780	#F06CC	-	65	53						
hCAT	Ext			-	72							
hCAT$	Ext			-	73							
hCOPYx	Ext			-	74							
hCREAT	Ext			-	75							
hDIDST	Ext			-	76							
hENTER	Ext			-	84							
hEXCPT	Ext			-	143							
hFINDF	Ext			-	89							
hFPROT	Ext			-	77							
hKYDF	Ext			-	93							
hPRTCL	Ext			-	80							
hPURGE	Ext			-	82							
hRDCBF	Ext			-	90							
hRDNBF	Ext			-	91							
hRENAM	Ext			-	83							
hVER$	Ext			-	66							
hWRCBF	Ext			-	92							
hZERPG	Ext			-	144							

Input Parameters

  Source file name is NZ&DIR::MS

  Listing file name is NZ/DIR:TI:ML::-1

  Object file name is NZ%DIR:TI:MS::-1

```
                                    111111
                          0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *
 2            *
 3            *       N   N  ZZZZZ   &      GGG   PPPP   RRRR
 4            *       N   N      Z  & &    G   G  P   P  R   R
 5            *       NN  N     Z   & &    G      P   P  R   R
 6            *       N N N    Z     &     G GGG  PPPP   RRRR
 7            *       N  NN   Z     & & &  G   G  P      R R
 8            *       N   N  Z      &  &   G   G  P      R  R
 9            *       N   N  ZZZZZ   && &   GGG   P      R   R
10            *
11            *
12                    TITLE  GENERAL ROUTINES <840106.1701>
13 F07C2              ABS    #F07C2        TIXHP6 address (fixed)
14            ****************************************************************
15            ****************************************************************
16            **
17            ** Name:        FRAME+ - Evaluate an HPIL message, return type
18            ** Name:        FRAME- - Evaluate a message, return type (not 3dat
19            **
20            ** Category:    PILUTL
21            **
22            ** Purpose:
23            **      Parses a frame
24            **
25            ** Entry:
26            **      C[6:0] contains the input frame from GET
27            **      ST[3:0] contains the HPIL handshake nibble
28            **
29            **      FRAME+: C[S] is the status nibble from DIAMOND
30            **
31            ** Exit:
32            **      Frame type in P:                                MNEMONIC:
33            **             0: ACKNOWLEDGE                            (pACK  )
34            **             1: CURRENT PIL STATE                      (pSTATE)
35            **             2: DIAGNOSTIC (TEST RESULTS)              (pDIAGR)
36            **             3: DIAGNOSTIC (LOCATION CONTENTS)         (pDIAGL)
37            **             4: ADDRESS                                (pADDR )
38            **             5: IFC RECEIVED (NOT SYS CONTROLLER)      (pIFC  )
39            **             6: ETO RECEIVED                           (pEOT  )
40            **             7: CONVERSATION HALTED (COUNT, NOT L)     (pHALTD)
41            **             8: TERMINATOR MATCH                       (pTERM )
42            **             9: ETE REVEIVED                           (pETE  )
43            **            10: UNRECOGNIZED TYPE                      (pUTYPE)
44            **            11: DATA/END FRAME                         (pDATA )
45            **            12: COMMAND RECEIVED                       (pCMD  )
46            **            13: READY FRAME                            (pRDY  )
47            **            14: IDY FRAME                              (pIDY  )
48            **            15: THREE BYTE DATA TRANSFER               (p3DATA)
49            **        If illegal frame or error, sets carry; else clears it
50            **
51            ** Calls:       None
52            **
53            ** Uses.......
54            **  Inclusive: C[S],P (C[S] only for FRAME+)
55            **
```

```
                ** Stk lvls:   0
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------    --------------------------------
                **  09/22/83      NZ          Updated documentation again
                **  01/03/83      NZ          Updated documentation
                **
                ********************************************************************
                ********************************************************************
F07C2 A46   =FRAME+ C=C+C   S               If carry, 3 byte data transfer
F07C5 80FF          CPEX    15
F07C9 560           GONC    FRAMEO          No carry...not 3 byte data
                *
                * Three byte data transfer!
                *
F07CC 20            P=      =p3DATA
F07CE 03            RTNCC
                *-
                *-
F07D0       =FRAME-
F07D0 0B    FRAMEO  CSTEX               Put the frame into status bits
F07D2 86B           ?ST=0   11          Is the MSB clear?
F07D5 41            GOYES   FROXXX      Yes!
                *
                * (1XXX XXXX XXXX) is data class
                *
                * (10XX XXXX XXXX) is DATA or END
                * (1100 XXXX XXXX) is COMMAND received
                * (1101 XXXX XXXX) is READY received
                * (111X XXXX XXXX) is IDY received
                *
F07D7 86A   FR1XXX  ?ST=0   10          Is bit 10 clear?
F07DA 20            GOYES   FR11XX      Yes...DATA or END
                *
                * Carry clear:
                *   (11XX XXXX XXXX) is COMMAND, READY, or IDY
                * Carry set:
                *   (10XX XXXX XXXX) is DATA or END
                *
F07DC 0B    FR11XX  CSTEX               Swap frame back into C[X]
F07DE 80D2          P=C     2           P is now the type!
F07E2 575           GONC    FREND       Go if COMMAND, READY, or IDY
                *FR10XX
                *
                * (10XX XXXX XXXX) is DATA or END
                *
F07E5 20            P=      =pDATA      Data/End
F07E7 03            RTNCC
                *-
                *-
F07E9       FROXXX
                *
                * (OXXX XXXX XXXX) is status, diagnostic, or address
```

```
111                *
112                * (0000 XXXX XXXX) is status message
113                * (0001 XXXX XXXX) is current state
114                * (001X XXXX XXXX) is diagnostic
115                * (01SS SSSP PPPP) is address
116                *
117 F07E9 86A              ?ST=0  10            Is it an address?
118 F07EC 80               GOYES  FR00XX        No!
119 F07EE 20               P=     =pADDR        Address
120 F07F0 0B               CSTEX
121 F07F2 03               RTNCC
122                *-
123                *-
124 F07F4          FR00XX
125                *
126                * (00XX XXXX XXXX) is either status or diagnostic class
127                *
128                * (0000 XXXX XXXX) is status message
129                * (0001 XXXX XXXX) is current state
130                * (001X XXXX XXXX) is diagnostic
131                *
132 F07F4 0B               CSTEX
133 F07F6 80D2             P=C    2
134 F07FA 880              ?P#    0
135 F07FD D3               GOYES  FREND         Current state or diagnostic
136                *
137                * (0000 ZZZZ XXXX) is status message if Z=0, else error
138                *
139 F07FF 21               P=     1
140 F0801 90E              ?C#0   P
141 F0804 34               GOYES  FRERR         Error!
142 F0806 0B               CSTEX
143 F0808 873              ?ST=1  3
144 F080B A3               GOYES  FRERRS        Unrecognized frame!
145                *
146                * (0000 0000 0XXX) is a status message...
147                *
148 F080D 0B               CSTEX
149 F080F 80D0             P=C    0             Decode it!
150 F0813 890              ?P=    =pACK
151 F0816 42               GOYES  FREND         NOP (Acknowledge)
152 F0818 883              ?P#    3             Is it ETE?
153 F081B 60               GOYES  FR00-3        No...
154 F081D 20               P=     =pETE         Yes...report it!
155 F081F 03               RTNCC
156                *-
157                *-
158 F0821 882      FR00-3  ?P#    2             Is it now an EOT?
159 F0824 60               GOYES  FR00-2        No...check further!
160 F0826 20               P=     =pEOT         Yes...
161 F0828 03               RTNCC
162                *-
163                *-
164 F082A 884      FR00-2  ?P#    4             Conversation halted?
165 F082D 60               GOYES  FR00-1        No...check further
```

```
 166 F082F 20          P=     =pHALTD
 167 F0831 03          RTNCC
 168           *-
 169           *-
 170 F0833 881 FR00-1  ?P#    1             IFC received?
 171 F0836 60          GOYES  FR00-0        Check further
 172 F0838 20          P=     =pIFC         Yes...set P to value!
 173 F083A 03  FREND   RTNCC
 174           *-
 175           *-
 176 F083C 885 FR00-0  ?P#    5             Terminator match?
 177 F083F 80          GOYES  FRERR         No...error!
 178 F0841 20          P=     =pTERM
 179 F0843 03          RTNCC
 180           *-
 181           *-
 182 F0845 0B  FRERRS  CSTEX                 Status back in ST, frame in C[X]
 183 F0847 20  FRERR   P=     =pUTYPE       This means unrecognized frame
 184 F0849 02          RTNSC
 185           ****************************************************************
 186           ****************************************************************
 187           **
 188           ** Name:       END - Clean up the loop
 189           ** Name:       ENDST - Clean up the loop, exit through NXTSTM
 190           ** Name:       ENDFN - Clean up the loop, preserve C[W] in R0
 191           ** Name:       UTLEND - Unaddress talkers&listeners, clean up
 192           **
 193           ** Category:   PILUTL
 194           **
 195           ** Purpose:
 196           **      Clean up after accessing a loop
 197           **
 198           ** Entry:
 199           **      MBOX^ points to the mailbox used by this routine
 200           **
 201           ** Exit:
 202           **      Carry clear:
 203           **        D0 at last mailbox used before call
 204           **        ENDST: Jumps to NXTSTM
 205           **        ENDFN: Restores value of C[W] (saved at entry)
 206           **        UTLEND: First unaddress talkers/listeners, then END
 207           **      Carry set:
 208           **        Error (P, C[0] are error code)
 209           **
 210           ** Calls:  END:GETMBX
 211           **         ENDST:END
 212           **        UTLEND:UNT,UNLPUT
 213           **         ENDFN:UTLEND
 214           **
 215           ** Uses.......
 216           **  Inclusive: C[W],D0,P,ST[3:0]
 217           **
 218           ** Stk lvls:   END: 0 <GETMBX>
 219           ** Stk lvls:   ENDST: 1 (END)
 220           ** Stk lvls:   UTLEND: 1 (UNT)(UNLPUT)<END>
```

```
                ** Stk lvls:   ENDFN: 2 (UTLEND)
                **
                ** History:
                **
                **    Date     Programmer          Modification
                **  --------   ----------   --------------------------------
                **  09/22/83      NZ        Updated documentation again
                **  01/03/83      NZ        Updated documentation
                **
                ****************************************************************
                ****************************************************************
F084B 7820      =ENDST  GOSUB  END
F084F 8C00      00      GOLONG =nXTSTM      Next basic statement!
                *-
                *-
F0855 108       =ENDFN  R0=C                Save value of C in R0!
F0858 7500              GOSUB  UTLEND
F085C 118               C=R0
F085F 01                RTN                 (Preserve carry!)
                *-
                *-
F0861 7210      =UTLEND GOSUB  Getmbx       Get the mailbox address
F0865 8E00      00      GOSUBL =UNT         Unaddress talkers
F086B 400               RTNC
F086E 7E84              GOSUB  UNLPUT       Unaddress listeners
F0872 821               XM=0                Clear XM flag (for statements)
F0875 01                RTN
                *-
                *-
F0877           =END
F0877 8C00      Getmbx  GOLONG =GETMBX      Return, D0 @ mailbox
                00
                ****************************************************************
                ****************************************************************
                **
                ** Name:       START - Set up entry conditions for the loop
                ** Name:       START+ - Set up loop information (loop # in C[S])
                ** Name:       START- - Set up loop (loop # in C[S], sReadd=1)
                **
                ** Category:   PILUTL
                **
                ** Purpose:
                **      Set up the loop, given the device specifier
                **
                ** Entry:
                **      D[3:0] contains the device address (if known).
                **          If the address is not known, D[B]=#1F/3F/5F/7F/9F
                **             #1F: (DevTyp) B[X] is the accessory ID
                **             #3F: (DevID)  B[W] is the device ID
                **             #5F: (VolLbl) B[W] is the volume label
                **             #7F: (Null)   B[W] is "don't care"
                **             #9F: (Loop)   B[W] is "don't care"
                **             D[2] is the sequence number for #1F and #3F
```

```
273           **      If D[X] is an address, bits 8 and 9 are the mailbox #
274           **      If D[X] is not an address, D[3] is the mailbox #
275           **
276           ** Exit:
277           **      Carry clear:
278           **        Device address in D[X] (+mailbox*1024)
279           **        D[S] is 0 if address given, 1 if device type,
280           **          2 if device ID, 3 if volume label, 4 if NULL,
281           **          5 if LOOP
282           **        Sets D0 to the HPIL mailbox
283           **        ST(sReadd) set if loop was readdressed, else clear
284           **      Carry set:
285           **        Error (P, C[0] are error code)
286           **
287           ** Calls:       SETLP,FNDCH-,GETDev,PUTGF-,PUTE,GETERR,GETST,
288           **              SFLAG?,RESTRT,GETMBX,SWAP01,I/OFND
289           **
290           ** Uses.......
291           **  Exclusive:      C[W],D[15   ],       D0,P,ST[4  ]
292           **  Inclusive: A[W],C[W],D[15:13],D[5:0],D0,P,ST[4:0]
293           **
294           ** Stk lvls:   3 (RESTRT)(FNDCH-)<GADDR>
295           **
296           ** Algorithm:
297           **      START: Derive loop # from D[X] (into C[S])     (SETLP)
298           **      START+:Set flag (sReadd) to not force readdressing
299           **      START-:Find mailbox, check for reset, OFFED   (FNDCH-)
300           **             Check if controller...if so, goto STARTn
301           **             Check if NULL, LOOP, or zero (if not, error)
302           **             goto START3
303           **            ----
304           **    (Controller)
305           **      STARTn:
306           **             If force readdressing (sReadd=1)
307           **               then send IFC to power up the loop
308           **               else send power up the loop message (NOP frame)
309           **      START5:Check if error powering up the loop    (GETERR)
310           **      START!:Get Diamond status bits
311           **             If sReadd=1 then goto START2
312           **             If loop is unconfigured (sUNCNF)
313           **               then
314           **                 If (supress readdress)=1 then goto START2
315           **                 Set all internal addresses=unknown (RESTRT)
316           **                 Set D0 to mailbox address          (GETMBX)
317           **             goto START3
318           **            ----
319           **    (Readdressing the loop)
320           **      START2:
321           **             Set all internal addresses=unknown     (RESTRT)
322           **             If (extended address flag=0) or
323           **                (an ASSIGNIO is active)
324           **               then readdress the loop, primary only
325           **               else readdress the loop, extended addresses
326           **             Send readdress message, get result     (PUTGF-)
327           **             If address not returned by Diamond then error
```

```
                **    (Check the device specifier)
                **      START3:If not (find device)
                **                then return (all OK)
                **                else goto GADDR (Get device address)
                **
                ** History:
                **
                **    Date     Programmer          Modification
                **  --------   ----------   --------------------------------
                **  09/22/83      NZ        Updated documentation
                **  08/02/83      NZ        Added check of =flNZ4 to check if
                **                          readdress the loop automatically
                **  06/03/83      NZ        Removed setting of IDY timeout
                **                          (now done in CHKSET)
                **  05/04/83      NZ        Removed redundant code to set the
                **                          device bit in LOOPST
                **  03/29/83      NZ        Added check of =flEXTD to decide
                **                          whether to use extended addresses
                **  03/15/83      NZ        Changed FNDMB-, CHKSTS to FNDCH-
                **  03/09/83      NZ        Changed code to call CHKSTS
                **  03/08/83      NZ        Changed call to FNDMBX to FNDMB-
                **  01/03/83      NZ        Updated documentation
                **
                ********************************************************************
                ********************************************************************
F087D           =START
                *
                * First set C[S] to be the mailbox #, minus 1
                *
F087D 8E00 00           GOSUBL =SETLP       Set loop # into C[S] from D[A]
                *
F0883 840       =START+ ST=0   =sReadd      START does NOT force readdress!
                *
                * Set DispOK bit false (Display is NOT set up on loop)
                *
F0886           =START-
                *
                * Get the mailbox address (search the device table for it)
                * (also clears DispOK and other bits in that nibble)
                *
F0886 7683              GOSUB  FNDCH-       Do all the above, FNDMBX,CHKSTS
F088A 400               RTNC
                *
                * Now DO points to the mailbox
                *
                * Check if I am the controller on this loop
                *
F088D 7F53              GOSUB  GETDev       Check if I am controller on loop
F0891 542               GONC   STARTn       I AM controller...continue
                *
                * If device is not "LOOP", "NULL" or zero then error, else
                * continue
                *
F0894 96B               ?D=0   B            Zero?
```

```
 382 F0897 B1          GOYES  STARTd       Yes...OK
 383 F0899 3100        LC(2)  =Null
 384 F089D 963         ?C=D   B            "NULL"?
 385 F08A0 21          GOYES  STARTd       Yes...OK!
 386 F08A2 3100        LC(2)  =Loop
 387 F08A6 963         ?C=D   B            "LOOP"?
 388 F08A9 90          GOYES  STARTd       Yes...OK!
 389           *
 390           * Error...Diamond is not controller and not LOOP, NULL, or 0
 391           *
 392 F08AB 300         LC(1)  =eBADMD      Illegal mode (not controller)
 393 F08AE 20          P=     =ePIL
 394 F08B0 02          RTNSC
 395           *-
 396           *-
 397 F08B2     STARTd
 398           *
 399           * I am in device mode!
 400           *
 401 F08B2 6AB0        GOTO   START3       Continue following controller
 402           *-
 403           *-
 404           *
 405           * I am controller...continue
 406           *
 407 F08B6     STARTn
 408           *
 409           * Diamond status in C[X]
 410           *
 411           * Power up the loop (check if need IFC or just mPULOP)
 412           *
 413 F08B6 870         ?ST=1  =sReadd      Force readdressing?
 414 F08B9 61          GOYES  START#       Yes...power up the loop with IFC
 415 F08BB 3100        LC(2)  =mPULOP      Power up the loop
 416 F08BF 7BB3        GOSUB  PUTGF-       Put it, GET, FRAME+
 417 F08C3 4C1         GOC    STARTS       Error...get the error message
 418 F08C6 890         ?P=    =pSTATE      Status message?
 419 F08C9 71          GOYES  STARTS       Yes...status message
 420 F08CB 64C0        GOTO   START5       No...unexpected frame
 421           *-
 422           *-
 423 F08CF 3500 START#  LC(6)  =mTAKEI      Take control with IFC
          0000
 424 F08D7 8E00        GOSUBL =PUTE
          00
 425 F08DD 400         RTNC                Carry if error
 426           *
 427           * Status message...check if error
 428           *
 429 F08E0 7500 STARTS  GOSUB  Geterr       Get error message
 430 F08E4 5A0         GONC   START!       If no carry, loop is UP!
 431 F08E7 02   STARtE  RTNSC               Error...exit with carry set
 432           *-
 433           *-
 434 F08E9 8C00 Geterr  GOLONG =GETERR      (P,C[0] are error, Carry set)
```

```
          00
435            *-
436            *-
437            *
438            * Now the loop is powered up!
439            *
440 F08EF 8E00 START!  GOSUBL =GETST        Get the Diamond status again
          00
441 F08F5 400          RTNC                If carry, ERROR!
442 F08F8 870  STARTO  ?ST=1  =sReadd       Force readdressing?
443 F08FB 62           GOYES  START2        Yes...do it!
444            *
445            * Check if loop needs to be readdressed (not done now)
446            *
447 F08FD 0B           CSTEX               Put Diamond status in ST bits
448 F08FF 860          ?ST=0  =sUNCNF      Is the loop unconfigured?
449 F0902 20           GOYES  START1       Set/Clear carry...
450 F0904 0B   START1  CSTEX               If carry is set, loop is OK!
451 F0906 466          GOC    START3
452 F0909 3100         LC(2)  =f1NZ4       Check if suppress auto readdress
453 F090D 7170         GOSUB  sflag?       Save D[A] in D0;SFLAG?;restore D
454 F0911 5F0          GONC   START2       Flag is clear...D0 readdress!
455 F0914 8E00         GOSUBL =RESTRT      Restart all devices! (unknown)
          00
456 F091A 795F         GOSUB  Getmbx       Flag is set...just get mailbox
457 F091E 5E4          GONC   START3       Go always
458            *-
459            *-
460 F0921 8E00 START2  GOSUBL =RESTRT      Set all devices to be restarted.
          00
461 F0927 850          ST=1   =sReadd      Indicate loop was readdressed!
462 F092A 3100         LC(2)  =f1EXTD      Check if extended addressing
463 F092E 7050         GOSUB  sflag?
464            *
465            * D[A] is the value of D0, which was saved there by SFLAG?
466            *
467 F0932 3100         LC(2)  (=mAUTOA)+1  Preset primary only!
468 F0936 522          GONC   STARTs       If flag is clear, use simple addr
469 F0939 8E00         GOSUBL =SWAP01      Swap D0, D1 to save D1
          00
470 F093F 3200         LC(3)  =bPILAI
          0
471 F0944 8E00         GOSUBL =i/OFND      Find the buffer
          00
472 F094A 8E00         GOSUBL =SWAP01      Restore D1 from D0
          00
473            *
474            * Now carry is SET if assignio buffer found
475            *
476 F0950 3100 STARTp  LC(2)  =mAUTOA       Loop needs to be reconfigured...
477 F0954 540          GONC   STARTs       If no carry, then no assignio
478 F0957 E6           C=C+1  A            If carry, then primary only
479            *
480 F0959 06   STARTs  RSTK=C              Save message on RSTK
481 F095B 781F         GOSUB  Getmbx       Get back the mailbox!!!
```

```
482 F095F 07          C=RSTK              Restore message
483 F0961 7913        GOSUB  PUTGF-       Put message, get last addr,decode
484 F0965 400         RTNC
485 F0968 880         ?P#    =pADDR       (address frame)
486 F096B 52          GOYES  START5
487 F096D AC3 START3  D=0    S            Set initial value of source flag
488 F0970 20          P=     0
489 F0972 310E        LCHEX  E0
490 F0976 0EFF        C=C!D  A            Check for address unknown
491 F097A B66         C=C+1  B            (address remains in D[3:0])
492 F097D 461         GOC    GADDR        Go if address unknown
493 F0980 03          RTNCC               Address is valid or 0
494           *-
495           *-
496 F0982 DF  sflag?  CDEX   A            Swap flag into D[A], D[A] to C
497 F0984 134         D0=C                Save D[A] in D0 (SFLAG? restores)
498 F0987 DB          C=D    A            Restore flag from D[A]
499 F0989 8D00 =sFLAG? GOVLNG =SFLAG?     Go to SFLAG? now
          000
500           *-
501           *-
502 F0990     START5
503 F0990 6741        GOTO   GADDRe       Unexpected frame error!
504           ********************************************************************
505           ********************************************************************
506           **
507           ** Name:       GADDR - Get the address of a device from loop
508           **
509           ** Category:   PILUTL
510           **
511           ** Purpose:
512           **      Get device address, given search information for the
513           **      device
514           **
515           ** Entry:
516           **      D0 points to the HPIL mailbox
517           **      D[B] is the search type (#1F,3F,5F,7F,9F)
518           **         #1F: (Device type) -B[B] is accessory ID
519           **         #3F: (Device ID)   -B[W] is device ID
520           **         #5F: (Volume label)-B[W] is the label
521           **         #7F: (Null)        -B[W] is "don't care"
522           **         #9F: (LOOP)        -B[W] is "don't care"
523           **      D[2] is the sequence number
524           **      D[3] is the loop number
525           **      D[S]=0 (for search type at exit)
526           **
527           ** Exit:
528           **      Carry clear:
529           **        HPIL handshake in ST[3:0]
530           **        Device address,(mailbox #)*1024 in D[X]
531           **        D[S] is search type (1=device type, 2=device ID,
532           **          3=volume label,4=NULL,5=LOOP)
533           **        D[3] is sequence number (was in D[2] at entry)
534           **      Carry set: P, C[S] are error code
535           **
```

```
536           ** Calls:      PUTGF+,UNLPUT,PUTC+,GETERR,GETID,PUTGF-,UNT,
537           **             TSTAT,SEEKA,DDT,TSTATA,READRG,ASRC4,MTYL,DDL
538           **
539           ** Uses.......
540           **  Exclusive: A[A],C[W],D[15:14],D[5:0],P
541           **  Inclusive: A[W],C[W],D[15:13],D[5:0],P,ST[3:0]
542           **             (If volume label, blankfills B[W],uses B[15:12])
543           **
544           ** Stk lvls:   3 (GETID)(TSTAT)(SEEKA)
545           **
546           ** Algorithm:
547           **      GADDR: if device type is not NULL then goto GADDR0
548           **             (Type=NULL)
549           **             set D[S] to DsNull-1
550           **      GADDRN:set address to zero
551           **             goto GADDR'
552           **            ----
553           **      GADDR0:if device type is not LOOP then goto GADDR1
554           **             (Type=LOOP)
555           **             set D[S] to DsLoop-1
556           **             goto GADDRN (set address=0, goto GADDR')
557           **            ----
558           **      GADDR1:if device type is not Acc ID then goto GADDR3
559           **             (Type=Accessory ID)
560           **             find that Acc ID (& sequence #)        (PUTGF+)
561           **             if not found then {Device Not Found}
562           **    (Device found, address message from Diamond in C[X])
563           **      GADDR':increment D[S] (search type)
564           **             set D[X]=address + (loop number)*1024 {bits 8&9}
565           **             set D[3]=sequence number (D[2] entry value)
566           **             return, all OK
567           **            ----
568           **    (Either Volume Label or Device ID)
569           **      GADDR3:determine length of word in B[W] by searching
570           **               from B[15] toward B[0], check for first non-
571           **               zero nibble (all unused nibbles of B[W]=0)
572           **             set D[14]=length (WP length)
573           **             if device type is not Device ID then goto GADDR6
574           **             (Type=Device ID)
575           **             make a copy of sequence number in D[5]
576           **             unaddress all listeners on loop        (UNLPUT)
577           **             reset Diamond current address           (PUTC+)
578           **             check for Diamond error (if so, exit)  (GETERR)
579           **             if loop is unaddressed then {Device Not Found}
580           **      GADDR4:get Device ID of the current device     (GETID)
581           **             if no response then goto GADDR5
582           **             if response matches requested ID (for given length
583           **               then
584           **                 decrement sequence # in D[5]
585           **                 if not right sequence number yet
586           **                   then goto GADDR5
587           **                   else
588           **                     set D[S]=0 (will be incremented twice)
589           **      GADDR&:        increment D[S]
590           **                     get current address           (PUTGF-)
```

```
591            **                    if not address then {Unexpected Frame}
592            **                    goto GADDR'
593            **                ----
594            **      GADDR5:increment current address              (PUTGF-)
595            **             if valid address then goto GADDR4
596            **             if end of addresses then {Device Not Found}
597            **               else {Unexpected Message}
598            **                ----
599            **      GADDR6:if device type <> Volume Label
600            **               then {Unexpected Frame}
601            **             (Type=Volume Label)
602            **             blankfill requested label (B[11:0])
603            **             set tape counter (D[4]) to first drive
604            **      GADDRv:find D[4]th tape drive                 (PUTGF+)
605            **             if not found then {Device Not Found}
606            **             check tape status                       (TSTAT)
607            **             if status <> all OK and status <> new tape
608            **               then goto GADDrn
609            **      GADDR7:seek sector zero on the tape            (SEEKA)
610            **             if seek error then goto GADDrn
611            **      GADDR8:read sector zero                          (DDT)
612            **             if read error then goto GADDrn         (TSTATA)
613            **             read 8 bytes from the tape             (READRG)
614            **             if tape is not LIF format then goto GADDrn
615            **             if tape volume label matches requested label
616            **               then
617            **                 set search type to 1 (will have 2 added)
618            **                 goto GADDR&
619            **    (rewind the tape, goto next tape)
620            **      GADDrn:rewind the current tape             (MTYL)(DDL)
621            **             increment tape counter (D[4])
622            **             if tape counter is >16 then {Device Not Found}
623            **             goto GADDRv
624            **
625            ** History:
626            **
627            **    Date     Programmer            Modification
628            **  --------   ----------   ---------------------------------
629            **  09/22/83      NZ        Updated documentation extensively
630            **  02/09/83      NZ        Added LOOP to valid lists
631            **  01/03/83      NZ        Updated documentation
632            **
633            ****************************************************************
634            ****************************************************************
635 F0994 DB   =GADDR  C=D    A             Copy D[2] (sequence #)
636            *
637            * Decode what type it is
638            *
639 F0996 3100         LC(2)  =Null         Is this a NULL assignment?
640 F099A 967          ?C#D   B
641 F099D F0           GOYES  GADDRO        Not NULL...continue
642            *
643            * NULL assignment!
644            *
645 F099F 2F           P=     15
```

```
646 F09A1 300          LC(1)  (=DsNull)-1   Code for NULL-1
647 F09A4 AC7  GADDRN  D=C    S
648 F09A7 D2           C=0    A             Clear "ADDRESS"
649 F09A9 503          GONC   GADDR'        Go always (Continue with coding)
650            *-
651            *-
652 F09AC 3100 GADDR0  LC(2)  =Loop          Is this LOOP?
653 F09B0 967          ?C#D   B
654 F09B3 A0           GOYES  GADDR1         Not LOOP...continue
655 F09B5 2F           P=     15
656 F09B7 300          LC(1)  (=DsLoop)-1   Code for LOOP-1
657 F09BA 59E          GONC   GADDRN         Go always
658            *-
659            *-
660 F09BD 3100 GADDR1  LC(2)  =DevTyp        Is this a device type?
661 F09C1 967          ?C#D   B
662 F09C4 34           GOYES  GADDR3         No...check further!
663            *
664            * Accessory ID...search for it!
665            *
666 F09C6 AE9          C=B    B             Type in C[B] for FIND Nth device
667 F09C9 23           P=     3
668 F09CB 300          LC(1)  =mFIND1       FIND Nth device, type M
669 F09CE 70B2         GOSUB  PUTGF+        Put message, get address,decode
670 F09D2 400          RTNC
671 F09D5 880          ?P#    =pADDR        Check if address frame
672 F09D8 B2           GOYES  GADDR2        Not address...error
673            *
674            * Entry with C[X] = Diamond "address" message
675            *
676 F09DA 0B   GADDR'  CSTEX                Clear the opcode bit!
677 F09DC 84A          ST=0   10
678 F09DF 0B           CSTEX
679            *
680            * Now address is in C[X], Seq # in D[2], loop # in D[3]
681            *
682 F09E1 B47          D=D+1  S             Set type flag=flag + 1
683 F09E4 DF           CDEX   A             Copy address to D[X]
684            *
685            * Now loop # in C[3], seq # in C[2]
686            *
687 F09E6 F2           CSL    A             Loop in C[4], seq in C[3]
688 F09E8 80D4         P=C    4
689 F09EC 80F2         CPEX   2             Loop in C[2], seq in C[3]
690 F09F0 AE2          C=0    B             Clear lower bits of C[X]
691 F09F3 A36          C=C+C  X
692 F09F6 A36          C=C+C  X             Now (loop #)*4 in C[XS]
693 F09F9 0E3F         C=C!D  X             C[X] is address+loop*1024
694 F09FD D7           D=C    A             D[X] is address,loop; D[3] is seq
695 F09FF 20           P=     0
696 F0A01 03           RTNCC                All done...exit, carry clear
697            *-
698            *-
699 F0A03 68A0 GADDR2  GOTO   GADDRn        Device not found
700            *-
```

```
                *-
F0A07           GADDR3
                *
                * Determine length of user-supplied string (store in D[14])
                *
F0A07 20                P=      0               First time through, P=15
                *
F0A09 979               ?B=0    W
F0A0C 90                GOYES   GADDR$          This SHOULD never happen!!!
                *
F0A0E 0D        GADDR?  P=P-1
F0A10 909               ?B=0    P
F0A13 BF                GOYES   GADDR?          Zero...continue checking
F0A15 AFF       GADDR$  CDEX    W
F0A18 80FE              CPEX    14              Put length in D[14]
F0A1C AFF               CDEX    W
                *
                * Now D[14] is user-supplied length
                *
F0A1F 20                P=      0
F0A21 3100              LC(2)   =DevID          Is this a device ID?
F0A25 963               ?C=D    B
F0A28 60                GOYES   GADDRd          Yes...device ID
F0A2A 66B0              GOTO    GADDR6          No...check volume label
                *-
                *-
                *
                * Device ID...search for the device!
                *
                * First unaddress all listeners on the loop
                *
F0A2E 7EC2 GADDRd       GOSUB   UNLPUT
F0A32 400               RTNC
                *
                * Now search the loop, asking each device its device ID
                *
                * Set current address to the start of the loop
                *
F0A35 3100              LC(2)   =MRSTCA         Reset current address to start
F0A39 8E00              GOSUBL  =PUTC+
      00
F0A3F 400               RTNC
                *
                * Read error message to clear the error flag (used to decide
                * when I'm done searching)
                *
F0A42 73AE              GOSUB   Geterr          Get error #
F0A46 400               RTNC                    If carry, had an error!
                *
                * Status bits are in C[X] now...check if addresses are valid
                *
F0A49 C6                C=C+C   A
F0A4B C6                C=C+C   A
F0A4D A66               C=C+C   B               Check if loop is unaddressed
F0A50 4B5               GOC     GADDRn          If so, say "Device Not Found"
```

```
755                *
756                * Addresses ARE valid...continue search
757                *
758 F0A53 F3           DSL    A            Now seq # in D[3], loop in D[4]
759 F0A55 DB           C=D    A            C[3] is seq #
760 F0A57 80D3         P=C    3            (Make a copy of seq # in D[5])
761 F0A5B 80C5         C=P    5            C[5] is now a copy of the seq #
762 F0A5F AB2          C=0    X            Clear the address field
763 F0A62 25           P=     5
764 F0A64 A97          D=C    WP           Copy back to D[5]
765                *
766                * Loop to check for the device ID!
767                *
768 F0A67 8E00 GADDR4  GOSUBL =GETID        Get ID of this device
            00
769 F0A6D 400          RTNC                Error (not "NOT READY")
770 F0A70 94B          ?D=0   S            ID response?
771 F0A73 B4           GOYES  GADDR5       No...try next
772 F0A75 970          ?A=B   W            Match exactly?
773 F0A78 E0           GOYES  GADDR-       Match...check for Nth item
774 F0A7A AFB          C=D    W            Not match...check user-given len
775 F0A7D 80DE         P=C    14
776 F0A81 914          ?A#B   WP
777 F0A84 A3           GOYES  GADDR5       Not a match...continue
778 F0A86 25   GADDR-  P=     5            Decrement copy of seq #
779 F0A88 A0F          D=D-1  P
780 F0A8B 523          GONC   GADDR5       Not Nth device...keep looking
781 F0A8E AC3          D=0    S            Set find flag=0 (+2 below=dev ID)
782
783                *
784                * Exact length match, Nth device!
785                *
786 F0A91 F7           DSR    A            Move loop # to D[3], seq to D[2]
787                *
788                * Now D[2] is sequence #, D[3] is loop #
789                *
790 F0A93 B47  GADDR&  D=D+1  S            Add 1 to current find flag
791 F0A96 20           P=     0
792 F0A98 3100         LC(2)  =mGETCA      Get current address
793 F0A9C 7ED1         GOSUB  PUTGF-
794 F0AA0 400          RTNC
795 F0AA3 880          ?P#    =pADDR       Not an address?
796 F0AA6 23           GOYES  GADDRe       Error...unexpected frame!
797                *
798                * GADDR' adds 1 to flag value!
799                *
800 F0AA8 613F         GOTO   GADDR'       Common exit code
801                *-
802                *-
803 F0AAC      GADDRn
804                *
805                * Device not found!
806                *
807 F0AAC 8E00         GOSUBL =UNT         Unaddress talkers on loop
            00
```

```
808 F0AB2 400          RTNC
809 F0AB5 20           P=     0
810 F0AB7 300          LC(1)  =eNOFND
811 F0ABA 20           P=     =ePIL
812 F0ABC 02           RTNSC                  Device not found!
813             *-
814             *-
815 F0ABE       GADDR5
816             *
817             * Not found yet...keep looking
818             *
819 F0ABE 20           P=     0
820 F0AC0 3100         LC(2)  =mINCCA         Increment current address
821 F0AC4 76B1         GOSUB  PUTGF-
822 F0AC8 400          RTNC                   Error
823 F0ACB 880          ?P#    =pADDR          Address?
824 F0ACE 50           GOYES  GADDRf          No...check frame further
825 F0AD0 569          GONC   GADDR4          Yes...poll the device for ID
826             *-
827             *-
828 F0AD3 890  GADDRf  ?P=    =pSTATE         Current state (error message)?
829 F0AD6 6D           GOYES  GADDRn          Yes...end of table (not found)
830             *
831             * Error...other than "NOT FOUND"
832             *
833 F0AD8       GADDRu                         Unknown device type...?
834 F0AD8 20   GADDRe  P=     0
835 F0ADA 300          LC(1)  =eUNEXP
836 F0ADD 20           P=     =ePIL
837 F0ADF 02           RTNSC
838             *-
839             *-
840 F0AE1       GADDR6
841             *
842             * Volume label?
843             *
844 F0AE1 3100         LC(2)  =VolLbl         Check if volume label
845 F0AE5 967          ?C#D   B
846 F0AE8 0F           GOYES  GADDRu          Unknown command
847             *
848             * Volume label!!!
849             *
850             * Find the 1st through 16th tapes, check the volume label
851             *
852             * First blank-fill the volume label!!!
853             * D[14] is first non-zero character in B...
854             *
855 F0AEA ADB            C=D    M               Get length from D[14] to C[14]
856 F0AED 3B02           LCASC  \       \       Blank-fill the volume label!
          0202
          0202
          02
857 F0AFB 80DE         P=C    14
858 F0AFF A99          C=B    WP
859 F0B02 AF5          B=C    W               Leave B[11:0] blank-filled
```

```
                        *
                        * D[4] is the current sequence # we are on!
                        *
F0B05 24                P=      4
F0B07 A83               D=0     P               Start with 1st device
F0B0A AB3               D=0     X               Clear address of tape for TSTAT
F0B0D 20        GADDRv  P=      0
F0B0F 3300              LC(4)   (=mFINDD)+#10 Find the Nth (Acc ID=16) drive
      00
F0B15 F7                DSR     A
F0B17 F7                DSR     A
F0B19 AAB               C=D     XS              Copy N from D[4]
F0B1C F3                DSL     A               Restore D[4]
F0B1E F3                DSL     A               Restore D[4]
F0B20 7E51              GOSUB   PUTGF+          Find Nth device, type #10
F0B24 400               RTNC                    Return if error, else check addr
F0B27 880               ?P#     =pADDR          Not address?
F0B2A 9A                GOYES   GADDRf          No...either "NOT FOUND" or error!
                        *
                        * Now current address is a tape device
                        *
F0B2C 8E00              GOSUBL  =TSTAT          Check tape status first
      00
F0B32 571               GONC    GADDR7          OK...seek record zero
                        *
                        * Check if "NEW TAPE" or other error!
                        *
F0B35 880               ?P#     =eTAPE
F0B38 00                RTNYES                  Not a tape error!
F0B3A 80F0              CPEX    0
F0B3E 880               ?P#     =eNEWTA         New tape?
F0B41 20                GOYES   GADDR+
F0B43 80F0      GADDR+  CPEX    0               Carry if NOT new tape
F0B47 4B5               GOC     GADDrn          Next item!
F0B4A D0        GADDR7  A=0     A
F0B4C 8E00              GOSUBL  =SEEKA          Seek record zero
      00
F0B52 590               GONC    GADDR8          OK!
F0B55 890       GADDr-  ?P=     =eTAPE          Tape error?
F0B58 B4                GOYES   GADDrn          Tape error...goto next item
F0B5A 02                RTNSC
                        *-
                        *-
F0B5C 22        GADDR8  P=      2
F0B5E 8E00              GOSUBL  =DDT            Read record zero
      00
F0B64 400               RTNC
F0B67 8E00              GOSUBL  =TSTATA         Check status
      00
F0B6D 47E               GOC     GADDr-          Not OK
F0B70 3500              LC(6)   (=mSDA)+8       Send 8 bytes
      0000
F0B78 8E00              GOSUBL  =READRG         Read register (A[W])
      00
F0B7E 400               RTNC
```

```
908                *
909                * D[S] is # characters in A[W] (rest is zero)
910                * (don't even check count...if less than 8 for any reason,
911                * will not match blankfilled volume label)
912                *
913                * Now A[3:0] is LIF ID (#8000), A[15:4] is volume label
914                *
915 F0B81 3308         LC(4)  #0080          LIF ID, byte-reversed
        00
916 F0B87 23           P=     3
917 F0B89 916          ?A#C   WP
918 F0B8C 71           GOYES  GADDrn         Tape not LIF...continue search
919                *
920                * This is an LIF tape...do the labels match???
921                *
922 F0B8E 7D83         GOSUB  ASRC4          Shift to A[11:0]
923 F0B92 2B           P=     11
924 F0B94 914          ?A#B   WP
925 F0B97 C0           GOYES  GADDrn         Label differs...try next!
926                *
927                * This volume label matches...found the device!
928                *
929 F0B99 AC3          D=0    S
930 F0B9C B47          D=D+1  S              Set find flag=1(+2) for vol label
931 F0B9F 63FE         GOTO   GADDR&         Get address, return!
932                *-
933                *-
934 F0BA3 7171 GADDrn  GOSUB  MTYL
935 F0BA7 400          RTNC
936 F0BAA 20           P=     =Rewind        Rewind the tape
937 F0BAC 8E00         GOSUBL =DDL
        00
938 F0BB2 400          RTNC
939 F0BB5 24           P=     4
940 F0BB7 B07          D=D+1  P              Increment tape counter
941 F0BBA 460          GOC    GADDR9         If carry, have searched 16 drives
942 F0BBD 6F4F         GOTO   GADDRv         Continue volume label search
943                *-
944                *-
945 F0BC1      GADDR9
946                *
947                * Device "NOT FOUND"
948                *
949 F0BC1 6AEE         GOTO   GADDRn         Device not found!
950                ****************************************************************
951                ****************************************************************
952                **
953                ** Name:       ATNCHK - Check if ATTN key has been hit twice
954                **
955                ** Category:   PILUTL
956                **
957                ** Purpose:
958                **      Check if ATNFLG has been decremented to "E" or less
959                **
960                ** Entry:
```

```
                **      None
                **
                ** Exit:
                **      Carry set: ATTN hit twice
                **      Carry clear: ATTN hit 0 or 1 times
                **
                ** Calls:      None
                **
                ** Uses.......
                **  Inclusive: C[S],P (P only if carry set)
                **
                ** Stk lvls:   1 (Internal push)
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **  --------   ----------   --------------------------------
                **  02/08/83      NZ        Wrote routine
                **
                ********************************************************************
                ********************************************************************
FOBC5 860       =ATNCHK ?ST=0  =Attn
FOBC8 62                GOYES  ATNCHc          Not aborting! (RTNCC)
                *
                * Attn set...check if ATNFLG true
                *
FOBCA 06                RSTK=C                 Save C[A] on RSTK
FOBCC 136               CDOEX                  Save DO in C[A]
FOBCF 1B00              DO=(5) =ATNFLG
      000
FOBD6 1564              C=DATO S
FOBDA 134               DO=C                   Restore DO
FOBDD 07                C=RSTK                 Restore C[A]
FOBDF 94A               ?C=0   S
FOBE2 C0                GOYES  ATNCHc          Not abort...(RTNCC)
FOBE4 B46               C=C+1  S               Check if "F"
FOBE7 460               GOC    ATNCHc          Yes...not abort (RTNCC)
FOBEA 20                P=     =eABORT         No...ABORT!
FOBEC 02                RTNSC
                *-
                *-
FOBEE 03        ATNCHc  RTNCC
                ********************************************************************
                ********************************************************************
                **
                ** Name:       GETDev - Get device status bit from LOOPST
                **
                ** Category:   PILUTL
                **
                ** Purpose:
                **      Indicate whether the last call to CHKSTS found Diamond
                **      in device or controller mode
                **
                ** Entry:
                **      None
```

```
1015                    **
1016                    ** Exit:
1017                    **      LOOPST in ST[3:0]
1018                    **      Carry set if device, clear if controller
1019                    **
1020                    ** Calls:      None
1021                    **
1022                    ** Uses.......
1023                    **  Inclusive: ST[3:0]
1024                    **
1025                    ** Stk lvls:   1 (internal push)
1026                    **
1027                    ** History:
1028                    **
1029                    **    Date      Programmer              Modification
1030                    **  --------    ----------    --------------------------------
1031                    **  03/17/83       NZ         Added code to save C[A] on RSTK
1032                    **  02/02/83       NZ         Added documentation
1033                    **
1034                    ********************************************************************
1035                    ********************************************************************
1036 F0BF0 06           =GETDev RSTK=C              Save C[A] on RSTK
1037 F0BF2 136                  CDOEX              Save DO in C[A]
1038 F0BF5 1B00                 DO=(5) =LOOPST
          000
1039 F0BFC 0B                   CSTEX
1040 F0BFE 15E0                 C=DATO 1           Read status into C[0]
1041 F0C02 0B                   CSTEX
1042 F0C04 134                  DO=C               Restore DO
1043 F0C07 07                   C=RSTK             Restore C[A]
1044 F0C09 870                  ?ST=1  (=Device)-8 (Device is set up for XS read)
1045 F0C0C 00                   RTNYES             Carry set if device
1046 F0C0E 03                   RTNCC              Carry clear if not device
1047                    ********************************************************************
1048                    ********************************************************************
1049                    **
1050                    ** Name:       CHKSTS - Check Diamond status, errors, etc
1051                    ** Name:       FNDCHK - Find a mailbox, CHKSTS
1052                    ** Name:       FNDCH- - Check OFFED, Find a mailbox, CHKSTS
1053                    **
1054                    ** Category:   EXCUTL
1055                    **
1056                    ** Purpose:
1057                    **      Check that the status is OK for messages (ie NOT in
1058                    **      manual mode), clear the error bit in Diamond, set/clear
1059                    **      bit for device/controller
1060                    **
1061                    ** Entry:
1062                    **      FNDCH-:C[S] is mailbox desired
1063                    **      FNDCHK:C[S] is mailbox desired
1064                    **      CHKSTS:DO points to mailbox
1065                    **
1066                    ** Exit:
1067                    **      Carry clear:
1068                    **        P=0, C[X] is Diamond status
```

```
1069            **        CHKSTS:D0 unchanged
1070            **        FNDCH-,FNDCHK:D0 points to mailbox
1071            **      Carry set: error (P, C[0] are the error #)
1072            **
1073            ** Calls:      GETHS2,CHKSET,GETERR,GETST,GETMBX
1074            **
1075            ** Uses.......
1076            **  Exclusive:      C[X],P
1077            **  Inclusive: A[W],C[W],P,ST[3:0], bit(Device) of LOOPST
1078            **
1079            ** Stk lvls:   2 (GETST)(GETERR)(CHKSET)(pushed status;GETMBX)
1080            **
1081            ** History:
1082            **
1083            **    Date     Programmer             Modification
1084            **  --------   ----------   ---------------------------------
1085            **  09/22/83      NZ        Updated documentation
1086            **  03/09/83      NZ        Wrote code and documentation
1087            **
1088            ****************************************************************
1089            ****************************************************************
1090 F0C10 8E00 =FNDCH- GOSUBL =FNDMB-       Find the mailbox, check it
         00
1091 F0C16 5D0          GONC   CHKSTS        If no error, continue
1092 F0C19 02           RTNSC
1093            *-
1094            *-
1095 F0C1B 8E00 =FNDCHK GOSUBL =FNDMBX       Find the mailbox first
         00
1096 F0C21 400          RTNC                 Error (not found)
1097 F0C24      =CHKSTS
1098 F0C24 8E00         GOSUBL =GETHS2
         00
1099 F0C2A 870          ?ST=1  =sMANUL       Manual mode?
1100 F0C2D 64           GOYES  CHKST+        Yes..Illegal mode (not auto mode)
1101 F0C2F 8E00         GOSUBL =CHKSET       Check if RESET: if so, initialize
         00
1102 F0C35 400          RTNC                 Error during initialize!
1103 F0C38 7DAC         GOSUB  Geterr        Get Gemstone status! (&clear err)
1104 F0C3C 5B0          GONC   CHKST.        If no carry, all is fine
1105 F0C3F 8E00         GOSUBL =GETST        If carry, get status bits
         00
1106 F0C45 400          RTNC                 Error!
1107 F0C48 0B   CHKST.  CSTEX                Put C[X] in the status bits
1108            *
1109            * Now check if I am the controller
1110            *
1111 F0C4A 870          ?ST=1  =sCONTR       Am I the controller on loop?
1112 F0C4D D2           GOYES  CHKSTn        Yes...done
1113            *
1114            * I am in device mode...set the Device bit of LOOPST
1115            *
1116 F0C4F 0B           CSTEX                Restore status bits,PILST-->C[X]
1117 F0C51 06           RSTK=C               Save PILST on RSTK
1118 F0C53 1B00         D0=(5) =LOOPST       Set =Device bit in LOOPST
```

```
            000
  1119 F0C5A 1562          C=DAT0 XS
  1120 F0C5E 0B            CSTEX
  1121 F0C60 850           ST=1   =Device       Set Device Status bit
  1122 F0C63 0B            CSTEX
  1123 F0C65 1542          DAT0=C XS            Write it out to LOOPST
  1124 F0C69 7A0C          GOSUB  Getmbx        Get DO back at mailbox
  1125 F0C6D 07            C=RSTK               Restore PILST from RSTK
  1126 F0C6F 03            RTNCC                Return, status in C[X]
  1127              *-
  1128              *-
  1129              *
  1130              * Error...Diamond is in manual mode!
  1131              *
  1132 F0C71 0B    CHKSTe  CSTEX
  1133 F0C73 300   CHKST+  LC(1)  =eBADMD       Illegal mode (not controller)
  1134 F0C76 20            P=     =ePIL
  1135 F0C78 02            RTNSC
  1136              *-
  1137              *-
  1138 F0C7A 0B    CHKSTn  CSTEX                Restore status bits
  1139 F0C7C 03            RTNCC
  1140              ****************************************************************
  1141              ****************************************************************
  1142              **
  1143              ** Name:       PUTGF- - CSL A,CSL A, call PUTC, GET, FRAME+
  1144              ** Name:       PUTGF+ - call PUTC, GET, FRAME+
  1145              ** Name:       PUTGF  - check carry, call GET, FRAME+
  1146              **
  1147              ** Category:   LOCAL
  1148              **
  1149              ** Purpose:
  1150              **      Save code by grouping commonly called subroutines
  1151              **
  1152              ** Entry:
  1153              **      DO points to mailbox
  1154              **      PUTGF-:C[B] is the message to send
  1155              **      PUTGF+:C[3:0] is the message to send
  1156              **      PUTGF: Carry set if previous error
  1157              **
  1158              ** Exit:
  1159              **      DO unchanged
  1160              **      Carry clear: P is frame type, C[X] is frame
  1161              **      Carry set: Error (P, C[0] are error code)
  1162              **
  1163              ** Calls:      PUTC,GET,<FRAME+>
  1164              **
  1165              ** Uses.......
  1166              **  Inclusive: C[W],P,ST[3:0]
  1167              **
  1168              ** Stk lvls:   1 (PUTC)(GET)
  1169              **
  1170              ** History:
  1171              **
  1172              **    Date     Programmer             Modification
```

```
1173            **  --------   ----------    --------------------------------
1174            **  09/22/83      NZ         Updated documentation
1175            **  02/28/83      NZ         Added PUTGF- entry point
1176            **  12/05/82      NZ         Added routine and documentation
1177            **
1178            ****************************************************************
1179            ****************************************************************
1180 F0C7E F2   =PUTGF- CSL    A
1181 F0C80 F2          CSL    A
1182 F0C82 7470 =PUTGF+ GOSUB  Putc          Put the message...
1183 F0C86 400  =PUTGF  RTNC
1184 F0C89 7580         GOSUB  Get           ...Get the response...
1185 F0C8D 400          RTNC
1186 F0C90 613B         GOTO   FRAME+        Exit through FRAME+!
1187            ****************************************************************
1188            ****************************************************************
1189            **
1190            ** Name:       GTYPE - Get the device type (Acc id) from loop
1191            **
1192            ** Category:   PILI/O
1193            **
1194            ** Purpose:
1195            **      Get the accessory id of a device (address in D[X])
1196            **
1197            ** Entry:
1198            **      DO points to the HPIL mailbox
1199            **      D[X] contains the address of the device to be checked
1200            **
1201            ** Exit:
1202            **      Carry clear:
1203            **        P=0
1204            **        Device type in A[B] (if 2 byte response, A[3:2] is
1205            **          first byte received, A[B] is second)
1206            **        If device does not respond to Acc ID, A[A]=0
1207            **      Carry set: error (P, C[0] are error code)
1208            **
1209            ** Calls:      YTML,PUTE,PUTGF
1210            **
1211            ** Uses.......
1212            **  Exclusive: A[A],C[W],P
1213            **  Inclusive: A[A],C[W],P,ST[3:0]
1214            **
1215            ** Stk lvls:   2 (YTML)(PUTGF)
1216            **
1217            ** History:
1218            **
1219            **    Date     Programmer             Modification
1220            **  --------   ----------    --------------------------------
1221            **  09/22/83      NZ         Updated documentation
1222            **  05/17/83      NZ         Rewrote to fix early EOT error
1223            **  01/03/83      NZ         Updated documentation
1224            **
1225            ****************************************************************
1226            ****************************************************************
1227 F0C94 7890 =GTYPE  GOSUB  YTML          YOU TALK, ME LISTEN
```

```
1228 F0C98 400          RTNC                    RETURN IF ERROR (CARRY SET)
1229 F0C9B D0           A=0     A               Clear value of acc id first
1230 F0C9D 3500         LC(6)   (=mSAI)+#2      LIMIT OF TWO BYTES
          0000
1231 F0CA5 8E00         GOSUBL  =PUTE           START ACCESSORY POLL
          00
1232 F0CAB 77DF GTYPE-  GOSUB   PUTGF           Do a GET, FRAME+
1233 F0CAF 400          RTNC                    If carry, error
1234            *
1235            * Now P is frame type
1236            *
1237 F0CB2 880          ?P#     =pDATA          Is this a data byte?
1238 F0CB5 51           GOYES   GTYPE2          No...check if EOT
1239 F0CB7 8AC          ?A#0    A
1240 F0CBA 20           GOYES   GTYPE0          Set carry if A#0 before this byte
1241 F0CBC F0   GTYPE0  ASL     A
1242 F0CBE F0           ASL     A               Save any previous data in A[3:2]
1243 F0CC0 AEA          A=C     B               Copy data byte to A[B]
1244 F0CC3 57E          GONC    GTYPE-          If no carry (A=0) then get next
1245 F0CC6 20   GTYPE1  P=      0               Reset P=0
1246 F0CC8 03           RTNCC                   Done...return!
1247            *-
1248            *-
1249 F0CCA 890  GTYPE2  ?P=     =pEOT           Is this an EOT frame?
1250 F0CCD 9F           GOYES   GTYPE1          Yes...done
1251 F0CCF 890          ?P=     =pSTATE         Is this an error message?
1252 F0CD2 C0           GOYES   GTYPE4          Yes...must mean error!
1253 F0CD4 20           P=      =eUNEXP         No...unexpected frame
1254 F0CD6 80C0 GTYPE3  C=P     0               Put the error message into C[0]
1255 F0CDA 20           P=      =ePIL
1256 F0CDC 02           RTNSC
1257            *-
1258            *-
1259 F0CDE 80D4 GTYPE4  P=C     4               Read error code
1260 F0CE2 880          ?P#     =eNORDY         Is it other than "NOT READY"?
1261 F0CE5 1F           GOYES   GTYPE3
1262 F0CE7 5ED          GONC    GTYPE1          Return, clear carry, P=0
1263            ****************************************************************
1264            ****************************************************************
1265            **
1266            ** Name:        ULYL - Unaddress listeners, address D[X] as Listen
1267            ** Name:        LISTEN - Address D[X] as listener
1268            **
1269            ** Category:    PILUTL
1270            **
1271            ** Purpose:
1272            **      Unaddress all listeners, address D[X] as listener
1273            **
1274            ** Entry:
1275            **      Desired listener address in D[X]
1276            **      D0 points to mailbox
1277            **
1278            ** Exit:
1279            **      Carry clear: OK, P=0
1280            **      Carry set: error (P=error #)
```

```
1281            **
1282            ** Calls:      PUTC
1283            **
1284            ** Uses.......
1285            **  Inclusive: C[W],P,ST[3:0]
1286            **
1287            ** Stk lvls:   1 (PUTC)
1288            **
1289            ** History:
1290            **
1291            **    Date      Programmer              Modification
1292            **  --------    ----------    ---------------------------------
1293            **  01/03/83      NZ          Updated documentation
1294            **
1295            ********************************************************************
1296            ********************************************************************
1297 FOCEA 7210 =ULYL   GOSUB  UNLPUT
1298 FOCEE 400         RTNC
1299 FOCF1 3300 =LISTEN LC(4)  =mADDRL       Address ( ) as listener
          00
1300 FOCF7 ABB  PUTC=D  C=D    X             Fill in ( )
1301 FOCFA 8C00 Putc    GOLONG =PUTC         Carry indicates return status
          00
1302            *-
1303            *-
1304 FOD00 20   =UNLPUT P=     0
1305 FOD02 3300         LC(4)  =mUNL         Unaddress all listeners
          00
1306 FOD08 61FF         GOTO   Putc
1307            *-
1308            *-
1309 FOD0C 8C00 =Putd   GOLONG =PUTD
          00
1310            *-
1311            *-
1312 FOD12 8C00 Get     GOLONG =GET
          00
1313            ********************************************************************
1314            ********************************************************************
1315            **
1316            ** Name:       MTYL - Unaddress listeners, me talk, D[X] listen
1317            ** Name:       MTYLL- Address me as talker, D[X] as listener
1318            **
1319            ** Category:   PILUTL
1320            **
1321            ** Purpose:
1322            **      Address me as talker, D[X] as listener
1323            **
1324            ** Entry:
1325            **      D[X] is the address of the device to be listener
1326            **      D0 points to mailbox
1327            **
1328            ** Exit:
1329            **      Carry clear: OK, P=0
1330            **      Carry set: error (P=error code)
```

```
1331            **
1332            ** Calls:      UNLPUT,LISTEN,<PUTC>
1333            **
1334            ** Uses.......
1335            **  Inclusive: C[W],P,ST[3:0]
1336            **
1337            ** Stk lvls:   1 (UNLPUT)(LISTEN)
1338            **
1339            ** History:
1340            **
1341            **    Date     Programmer              Modification
1342            **  --------   ----------    ---------------------------------
1343            **  01/03/83      NZ         Updated documentation
1344            **
1345            ****************************************************************
1346            ****************************************************************
1347 F0D18 74EF =MTYL   GOSUB  UNLPUT        Unaddress all listeners
1348 F0D1C 400         RTNC                 RETURN IF ERROR (CARRY SET)
1349 F0D1F 7ECF =MTYLL  GOSUB  LISTEN        Address D[X] as listener
1350 F0D23 400         RTNC                 RETURN IF ERROR (CARRY SET)
1351 F0D26 3300 =MTYLC  LC(4)  (=MADDRM)+#4  / ADDRESS ME AS TALKER
          00
1352 F0D2C 6DCF         GOTO   Putc          \   (carry=status)
1353            ****************************************************************
1354            ****************************************************************
1355            **
1356            ** Name:       YTML - "You" (D[X]) talk, "me" listen
1357            **
1358            ** Category:   PILUTL
1359            **
1360            ** Purpose:
1361            **      Address D[X] as talker, me as listener
1362            **
1363            ** Entry:
1364            **      DO points to mailbox
1365            **      D[X] contains the address of the device to be talker
1366            **
1367            ** Exit:
1368            **      Carry clear: P=0
1369            **      Carry set: Error # in P
1370            **
1371            ** Calls:      UNLPUT,PUTC,<PUTC=D>
1372            **
1373            ** Uses.......
1374            **  Inclusive: C[W],P,ST[3:0]
1375            **
1376            ** Stk lvls:   1 (UNLPUT)(PUTC)
1377            **
1378            ** History:
1379            **
1380            **    Date     Programmer              Modification
1381            **  --------   ----------    ---------------------------------
1382            **  01/03/83      NZ         Updated documentation
1383            **
1384            ****************************************************************
```

```
1385            ****************************************************************
1386 F0D30 7CCF =YTML   GOSUB  UNLPUT        Unaddress all listeners
1387 F0D34 400          RTNC                 Return if error (carry set)
1388 F0D37 3300 =YTMLL  LC(4)  (=mADDRM)+#2  Address me as listener
          00
1389 F0D3D 79BF         GOSUB  Putc
1390 F0D41 400          RTNC                 Return if error (carry set)
1391 F0D44 3300 =TALK   LC(4)  =mADDRT
          00
1392 F0D4A 6CAF         GOTO   PUTC=D        Address D[X] as talker
1393            ****************************************************************
1394            ****************************************************************
1395            **
1396            ** Name:       PRMSGA - Output message from C (uses A)
1397            **
1398            ** Category:   PILI/O
1399            **
1400            ** Purpose:
1401            **      Output message from C (ASCII) (use A[W] to store it)
1402            **
1403            ** Entry:
1404            **      C[W] has an ASCII string, C[B] is the first character
1405            **      Message is terminated by a #00 character
1406            **      D0 points to mailbox
1407            **
1408            ** Exit:
1409            **      Carry clear: OK, P=0
1410            **      Carry set: error (P,C[0] are error code)
1411            **
1412            ** Calls:      PUTD
1413            **
1414            ** Uses.......
1415            **  Inclusive: A[W],C[W],ST[3:0]
1416            **
1417            ** Stk lvls:   1 (PUTD)
1418            **
1419            ** Algorithm:
1420            **      PRMSGA:Copy C[W] to A[W]
1421            **      PRMSG1:shift A[W] right twice (next char in A[B] now)
1422            **             output the character in C[B]            (PUTD)
1423            **             if next character (A[B]) <> #00 then goto PRMSG1
1424            **             return
1425            **
1426            ** History:
1427            **
1428            **    Date     Programmer          Modification
1429            **  --------   ----------   --------------------------------
1430            **  01/03/83      NZ        Updated documentation
1431            **
1432            ****************************************************************
1433            ****************************************************************
1434 F0D4E AFA  =PRMSGA A=C    W            First byte is still in C[B]
1435 F0D51 BF4  PRMSG1  ASR    W            Get next char into A[B]
1436 F0D54 BF4          ASR    W
1437 F0D57 71BF         GOSUB  Putd         Output the character
```

```
1438 F0D5B 400          RTNC                    Return if error (carry set)
1439 F0D5E D6           C=A     A               Get next byte
1440 F0D60 96E          ?C#0    B               Is this the end (NULL byte)?
1441 F0D63 EE           GOYES   PRMSG1          No...output it!
1442 F0D65 01           RTN                     Yes...return, carry clear
1443            ****************************************************************
1444            ****************************************************************
1445            **
1446            ** Name:        DTOH - Convert from decimal to HEX
1447            **
1448            ** Category:    PILUTL
1449            **
1450            ** Purpose:
1451            **      Convert value in A[A] from decimal to hex
1452            **
1453            ** Entry:
1454            **      A[A] contains the BCD value
1455            **      A[S] contains the sign of the value (for exit only)
1456            **
1457            ** Exit:
1458            **      Hex value in C[A], sign in C[S] (copied from A[S])
1459            **      P=0, carry clear
1460            **
1461            ** Calls:       None
1462            **
1463            ** Uses.......
1464            **  Inclusive: A[A],B[A],C[A],P
1465            **
1466            ** Stk lvls:    0
1467            **
1468            ** History:
1469            **
1470            **    Date      Programmer              Modification
1471            **  --------    ----------    ---------------------------------
1472            **  01/03/83      NZ          Updated documentation
1473            **
1474            ****************************************************************
1475            ****************************************************************
1476 F0D67 04   =DTOH    SETHEX
1477 F0D69 20           P=      0
1478 F0D6B 3401         LC(5)   10000
           720
1479 F0D72 D1           B=0     A
1480 F0D74 24           P=      4
1481 F0D76 908  DTOH0   ?A=0    P
1482 F0D79 A0           GOYES   DTOH1
1483 F0D7B C1           B=B+C   A
1484 F0D7D A0C          A=A-1   P
1485 F0D80 55F          GONC    DTOH0           Go always
1486            *-
1487            *-
1488 F0D83 0D   DTOH1   P=P-1
1489 F0D85 136          CDOEX                   Use DO to set value!
1490 F0D88 883          ?P#     3
1491 F0D8B B0           GOYES   DTOH2
```

```
1492 F0D8D 1A8E          D0=(4) 1000
          30
1493 F0D93 591           GONC   DTOH4          Go always
1494               *-
1495               *-
1496 F0D96 882  DTOH2    ?P#    2
1497 F0D99 B0            GOYES  DTOH3
1498 F0D9B 1A46          D0=(4) 100
          00
1499 F0DA1 5B0           GONC   DTOH4          Go always
1500               *-
1501               *-
1502 F0DA4 881  DTOH3    ?P#    1
1503 F0DA7 60            GOYES  DTOH4
1504 F0DA9 19A0          D0=(2) 10
1505               *
1506 F0DAD 136  DTOH4    CDOEX
1507 F0DB0 55C           GONC   DTOH0          If carry clear, not done yet
1508               *
1509               * Done! (P=0, carry set)
1510               *
1511 F0DB3 D9            C=B    A
1512 F0DB5 C2            C=C+A  A              Now HEX result in C[A]!
1513 F0DB7 AC6           C=A    S              (Copy sign from A[S])
1514 F0DBA 03            RTNCC
1515               ************************************************************
1516               ************************************************************
1517               **
1518               ** Name:       HTOD - Convert C[B] value from hex to decimal
1519               **
1520               ** Category:   PILUTL
1521               **
1522               ** Purpose:
1523               **      Convert C[B] from hex into decimal, use only B,C,P
1524               **
1525               ** Entry:
1526               **      C[B] contains a HEX value
1527               **
1528               ** Exit:
1529               **      Decimal value in B[X]
1530               **      Decimal mode set!
1531               **      Carry set, P=3
1532               **
1533               ** Calls:      None
1534               **
1535               ** Uses.......
1536               **  Inclusive: B[A],C[A],P
1537               **
1538               ** Stk lvls:   0
1539               **
1540               ** History:
1541               **
1542               **    Date     Programmer             Modification
1543               **  --------   ----------    ---------------------------------
1544               **  01/03/83      NZ         Updated documentation
```

```
1545                **
1546                ****************************************************************
1547                ****************************************************************
1548 F0DBC D1      =HTOD   B=0    A             Clear destination register
1549 F0DBE F2              CSL    A
1550 F0DC0 F2              CSL    A             Save digits in C[3:2]
1551 F0DC2 20              P=     0
1552 F0DC4 05              SETDEC
1553                *
1554                * Loop for the case of A-F
1555                *
1556 F0DC6 A2E   HTOD1    C=C-1  XS            Is the least sig. digit zero?
1557 F0DC9 470             GOC    HTOD2         Yes...next digit
1558 F0DCC E5              B=B+1  A             No...increment result
1559 F0DCE 57F             GONC   HTOD1         Go always
1560                *-
1561                *-
1562 F0DD1 3261  HTOD2    LCHEX  016           Now the digit value is 16(DEC)
          0
1563 F0DD6 23              P=     3             Point to the other digit
1564 F0DD8 A0E   HTOD3    C=C-1  P             Is the digit zero yet?
1565 F0DDB 400             RTNC                Yes...done!
1566 F0DDE A31             B=B+C  X             No...add another 16!
1567 F0DE1 56F             GONC   HTOD3         Go always
1568                ****************************************************************
1569                ****************************************************************
1570                **
1571                ** Name:       HTODX - Convert A[W] from HEX to decimal
1572                **
1573                ** Category:   PILUTL
1574                **
1575                ** Purpose:
1576                **      Convert A[W] from HEX to DECIMAL
1577                **
1578                ** Entry:
1579                **      A[W] contains the HEX value
1580                **
1581                ** Exit:
1582                **      Carry clear: Decimal value in B[W], P#0
1583                **      Carry set: Error (range error) (P=Error #)
1584                **
1585                ** Calls:      None
1586                **
1587                ** Uses.......
1588                **  Inclusive: A[W],B[W],C[W],P
1589                **
1590                ** Stk lvls:   0
1591                **
1592                ** History:
1593                **
1594                **    Date     Programmer              Modification
1595                **  --------   ----------    ---------------------------------
1596                **  01/03/83      NZ         Updated documentation
1597                **
1598                ****************************************************************
```

```
1599            ****************************************************************
1600 F0DE4 AF1 =HTODX  B=0    W
1601 F0DE7 AF2         C=0    W
1602 F0DEA 20          P=     0
1603 F0DEC 301         LC(1)  1
1604 F0DEF 05          SETDEC
1605 F0DF1 A0C HTODX1  A=A-1  P             Is this digit zero yet?
1606 F0DF4 480         GOC    HTODX2       Yes...continue with next!
1607 F0DF7 A71         B=B+C  W             No...add HEX place value to B[W]
1608 F0DFA 56F         GONC   HTODX1       Go always!
1609            *-
1610            *-
1611 F0DFD A80 HTODX2  A=0    P             Clear digit when done with it!
1612 F0E00 97C         ?A#0   W             Done with whole word?
1613 F0E03 60          GOYES  HTODX3       No...continue
1614            *
1615            * Carry clear if fall through
1616            *
1617 F0E05 04  HTODXr  SETHEX               Done...return in HEX mode!
1618 F0E07 01          RTN                  (Carry is result)
1619            *-
1620            *-
1621 F0E09 0C  HTODX3  P=P+1                Go to next digit
1622 F0E0B 411         GOC    HTODX4       Error!
1623 F0E0E A76         C=C+C  W             Do a multiply by 16 in DEC mode
1624 F0E11 A76         C=C+C  W
1625 F0E14 A76         C=C+C  W
1626 F0E17 A76         C=C+C  W
1627 F0E1A 56D         GONC   HTODX1
1628 F0E1D 20  HTODX4  P=     =eRANGE       Range error (Overflow)
1629 F0E1F 45E         GOC    HTODXr       Go always
1630            ****************************************************************
1631            ****************************************************************
1632            **
1633            ** Name:       A-MULT - Multiply A[A] by C[A], result in A[9:0]
1634            **
1635            ** Category:   MTHUTL
1636            **
1637            ** Purpose:    Multiply 20-bit hex integers
1638            **
1639            ** Entry:
1640            **      A[A], C[A] are the operands
1641            **      If HEXMODE, does HEX multiply; if DECMODE, DECIMAL mult
1642            **
1643            ** Exit:
1644            **      P has been preserved
1645            **      A[9:0] = product
1646            **      Carry set
1647            **
1648            ** Uses.......
1649            **  Inclusive: A[W],B[W],C[W]
1650            **
1651            ** Stk lvls:   0
1652            **
1653            **    Date     Programmer             Modification
```

```
1654            **  --------   ----------   --------------------------------
1655            **  09/22/83       NZ         Updated documentation
1656            **  12/06/82       NZ         Changed result to A[9:0]
1657            **  01/01/00       SA         Wrote original (mainframe)
1658            **
1659            ****************************************************************
1660            ****************************************************************
1661 F0E22 AF1  =A-MULT B=0     W
1662 F0E25 DC           ABEX    A              B[A] is multiplicand, B[15:5]=0
1663 F0E27 AF0          A=0     W              Clear result register
1664 F0E2A 8AA          ?C=0    A              Zero multiplier?
1665 F0E2D 00           RTNYES                 Yes...return, carry set
1666 F0E2F 80FE         CPEX    14             Save P in C[14]
1667 F0E33 20           P=      0
1668 F0E35 570          GONC    M-STRT         Go always
1669            *-
1670            *-
1671 F0E38 0C    NXTDGT P=P+1                  Go to next digit
1672 F0E3A BF1          BSL     W              Shift multiplicand left one digit
1673 F0E3D B8A   M-STRT C=-C    P              Zero digit?
1674 F0E40 57F          GONC    NXTDGT         Yes...go to next digit
1675            *
1676 F0E43 A70   ADCYCL A=A+B   W              ADD MULTIPLICAND TO RESULT
1677 F0E46 431          GOC     OVFLOW         ***This will NEVER happen!***
1678 F0E49 B06          C=C+1   P              Increment digit
1679 F0E4C 56F          GONC    ADCYCL         No carry=not done...repeat ADCYCL
1680 F0E4F 8AE          ?C#0    A              More digits remaining?
1681 F0E52 6E           GOYES   NXTDGT         Yes...go to next digit
1682 F0E54 80DE         P=C     14             No...restore P from C[14]
1683 F0E58 02           RTNSC                  Return with carry set...OK
1684            *-
1685            *-
1686 F0E5A AF0  OVFLOW  A=0     W              ***This code will never be used***
1687 F0E5D A7C          A=A-1   W              ***      .       .       ***
1688 F0E60 80DE         P=C     14             ***              .       ***
1689 F0E64 03           RTNCC                  ***      .       .       ***
1690            ****************************************************************
1691            ****************************************************************
1692            **
1693            ** Name:       UCRANG - Convert to upper case, check if [A-Z]
1694            ** Name:       CONVUC - Convert to upper case
1695            ** Name:       RANGE - Check if in given range
1696            ** Name:       RANGEN - Check if in [0-9]
1697            ** Name:       RANGEA - Check if in [A-Z]
1698            **
1699            ** Category:   PILUTL
1700            **
1701            ** Purpose:
1702            **      A[B] is item to work with:
1703            **      UCRANG: Determine if letter, convert to upper case
1704            **      CONVUC: Convert to upper case (if lower case)
1705            **      RANGE:  Determine if in specified range of characters
1706            **      RANGEN: Check if in [0-9]
1707            **      RANGEA: Check if in [A-Z]
1708            **
```

```
1709            ** Entry:
1710            **      A[B] contains the character to be checked
1711            **      P=0, HEXMODE
1712            **
1713            ** Exit:
1714            **      P=0
1715            **      Carry set if not in range
1716            **      Carry clear if in range
1717            **
1718            ** Calls:(UCRANG):          RANGEA,<CONVUC>
1719            ** Calls:(CONVUC):          RANGE
1720            ** Calls:(RANGE):           None
1721            **
1722            ** Uses.......
1723            **  Inclusive: C[A] (CONVUC also changes A[B] if in [a-z]
1724            **
1725            ** Stk lvls (UCRANG):       1 (RANGEA)<CONVUC>
1726            ** Stk lvls (CONVUC):       1 (RANGE)
1727            ** Stk lvls (RANGE):        0
1728            **
1729            ** History:
1730            **
1731            **    Date     Programmer          Modification
1732            **  --------   ----------   ---------------------------------
1733            **  01/03/83      NZ        Updated documentation
1734            **
1735            ****************************************************************
1736            ****************************************************************
1737 F0E66 7910 =UCRANG GOSUB  RANGEA        Check if in [A-Z]
1738 F0E6A 500          RTNNC               If carry clear, Done!
1739            *
1740            * Fall through to convert to upper case
1741            *
1742 F0E6D 3316 =CONVUC LCASC  \za\
          A7
1743 F0E73 7C10         GOSUB  RANGE
1744 F0E77 400          RTNC
1745 F0E7A 3102         LCHEX  20
1746 F0E7E B6A          A=A-C  B
1747 F0E81 03           RTNCC
1748            *-
1749            *-
1750 F0E83 3314 =RANGEA LCASC  \ZA\
          A5
1751 F0E89 6900         GOTO   RANGE
1752            *-
1753            *-
1754 F0E8D 3303 =RANGEN LCASC  \90\
          93
1755 F0E93 9E2  =RANGE  ?A<C   B
1756 F0E96 00           RTNYES
1757 F0E98 F6           CSR    A
1758 F0E9A BB6          CSR    X
1759 F0E9D B62          C=C-A  B
1760 F0EA0 01           RTN
```

```
1761            ****************************************************************
1762            ****************************************************************
1763            **
1764            ** Name:       GETALR - Get data into A[W] from @D1,left>right
1765            **
1766            ** Category:   PILUTL
1767            **
1768            ** Purpose:
1769            **      Read data from @ D1 into A[W], from A[15:14] to A[B]
1770            **
1771            ** Entry:
1772            **      D1 points to the data in RAM
1773            **      P is a count of bytes to be read into A[W]
1774            **      Bytes are to be entered with the last byte in A[B]
1775            **
1776            ** Exit:
1777            **      "P" data bytes in A[W]
1778            **      P=0
1779            **
1780            ** Calls:      None
1781            **
1782            ** Uses.......
1783            **  Inclusive: A[W],C[B],D1,P
1784            **
1785            ** Stk lvls:   0
1786            **
1787            ** History:
1788            **
1789            **    Date      Programmer              Modification
1790            **  --------    ----------    ---------------------------------
1791            **  01/03/83      NZ          Updated documentation
1792            **
1793            ****************************************************************
1794            ****************************************************************
1795 F0EA2 14F  =GETALR C=DAT1 B
1796 F0EA5 171         D1=D1+ 2
1797 F0EA8 BF0  =ALRNOG ASL    W
1798 F0EAB BF0         ASL    W
1799 F0EAE AEA         A=C    B
1800 F0EB1 0D          P=P-1
1801 F0EB3 880         ?P#    0
1802 F0EB6 CE          GOYES  GETALR
1803 F0EB8 01          RTN
1804            ****************************************************************
1805            ****************************************************************
1806            **
1807            ** Name:       PUTARL - Put data from A[W] (Right to left)
1808            ** Name:       PUTALR - Put data from A[W] (Left to right)
1809            **
1810            ** Category:   PILUTL
1811            **
1812            ** Purpose:
1813            **      Output data from A[W] to the HPIL loop
1814            **
1815            ** Entry:
```

```
1816            **      D0 points to mailbox
1817            **      I am talker on loop
1818            **      P is a count of bytes to be output from A[W]
1819            **      PUTARL outputs bytes starting with A[B]
1820            **      PUTALR outputs bytes starting with A[15:14]
1821            **
1822            ** Exit:
1823            **      Carry clear: P=0, all OK
1824            **      Carry set: error (P, C[0] are error code)
1825            **
1826            ** Calls:      PUTD
1827            **
1828            ** Uses.......
1829            **  Exclusive: A[W],C[A],P
1830            **  Inclusive: A[W],C[W],P,ST[3:0]
1831            **
1832            ** Stk lvls:   1 (PUTD)
1833            **
1834            ** History:
1835            **
1836            **    Date      Programmer              Modification
1837            **  --------    ----------    --------------------------------
1838            **  01/03/83      NZ          Updated documentation
1839            **
1840            ****************************************************************
1841            ****************************************************************
1842 F0EBA D6   =PUTARL C=A    A
1843 F0EBC 814          ASRC
1844 F0EBF 814          ASRC
1845            *
1846            * Put A[W] from right to left, no shift
1847            *
1848 F0EC2 764E =ARLNOS GOSUB  Putd
1849 F0EC6 400          RTNC                    Return if error (carry set)
1850 F0EC9 0D           P=P-1
1851 F0ECB 880          ?P#    0
1852 F0ECE CE           GOYES  PUTARL
1853 F0ED0 01           RTN                     Done!
1854            *-
1855            *-
1856 F0ED2 810  =PUTALR ASLC
1857 F0ED5 810          ASLC
1858 F0ED8 D6           C=A    A
1859            *
1860            * Put A[W] from left to right, no shift
1861            *
1862 F0EDA 7E2E =ALRNOS GOSUB  Putd
1863 F0EDE 400          RTNC                    Return if error (carry set)
1864 F0EE1 0D           P=P-1
1865 F0EE3 880          ?P#    0
1866 F0EE6 CE           GOYES  PUTALR
1867 F0EE8 01           RTN                     Done!
1868            ****************************************************************
1869            ****************************************************************
1870            **
```

```
1871            ** Name:       PUTDX - Output multiple data bytes (P is count)
1872            **
1873            ** Category:   PILI/O
1874            **
1875            ** Purpose:
1876            **      Output data to the loop: first the contents of C[B],
1877            **      then P-1 zero bytes
1878            **
1879            ** Entry:
1880            **      D0 points to mailbox
1881            **      I am talker
1882            **      P contains the total number of bytes to send
1883            **
1884            ** Exit:
1885            **      P=0
1886            **      Carry set if error (P is error #)
1887            **
1888            ** Calls:      PUTD
1889            **
1890            ** Uses.......
1891            **  Exclusive: C[A],P
1892            **  Inclusive: C[W],P,ST[3:0]
1893            **
1894            ** Stk lvls:   1 (PUTD)
1895            **
1896            ** History:
1897            **
1898            **    Date     Programmer              Modification
1899            **  --------   ----------    --------------------------------
1900            **  01/03/83      NZ         Updated documentation
1901            **
1902            ******************************************************************
1903            ******************************************************************
1904 F0EEA 7E1E =PUTDX  GOSUB  Putd
1905 F0EEE D2          C=0    A
1906 F0EF0 400         RTNC                  Return if error (carry set)
1907 F0EF3 0D          P=P-1
1908 F0EF5 880         ?P#    0
1909 F0EF8 2F          GOYES  PUTDX
1910 F0EFA 01          RTN                   Done!
1911            ******************************************************************
1912            ******************************************************************
1913            **
1914            ** Name:       ASLCn - Shift the A register n nibbles LEFT
1915            ** Name:       ASRCn - Shift the A register n nibbles RIGHT
1916            **
1917            ** Category:   PILUTL
1918            **
1919            ** Purpose:
1920            **      Shift the A register by a given number of nibbles
1921            **
1922            ** Entry:
1923            **      None
1924            **
1925            ** Exit:
```

```
1926            **      A[W] is rotated the given # of nibbles
1927            **
1928            ** Calls:      None
1929            **
1930            ** Uses.......
1931            **  Inclusive: A[W] (shifted as per instructions)
1932            **
1933            ** Stk lvls:   0
1934            **
1935            ** NOTE: Does not alter P or carry!!!
1936            **
1937            ** History:
1938            **
1939            **    Date     Programmer              Modification
1940            **  --------   ----------    ---------------------------------
1941            **  01/03/83      NZ         Updated documentation
1942            **
1943            ********************************************************************
1944            ********************************************************************
1945 F0EFC      =ASRC8
1946 F0EFC 810  =ASLC8  ASLC
1947 F0EFF      =ASRC9
1948 F0EFF 810  =ASLC7  ASLC
1949 F0F02      =ASRC10
1950 F0F02 810  =ASLC6  ASLC
1951 F0F05      =ASRC11
1952 F0F05 810  =ASLC5  ASLC
1953 F0F08      =ASRC12
1954 F0F08 810  =ASLC4  ASLC
1955 F0F0B      =ASRC13
1956 F0F0B 810  =ASLC3  ASLC
1957 F0F0E      =ASRC14
1958 F0F0E 810  =ASLC2  ASLC
1959 F0F11      =ASRC15
1960 F0F11 810  =ASLC1  ASLC
1961 F0F14 01           RTN
1962            *-
1963            *-
1964 F0F16      =ASRC7
1965 F0F16 814  =ASLC9  ASRC
1966 F0F19      =ASRC6
1967 F0F19 814  =ASLC10 ASRC
1968 F0F1C      =ASRC5
1969 F0F1C 814  =ASLC11 ASRC
1970 F0F1F      =ASRC4
1971 F0F1F 814  =ASLC12 ASRC
1972 F0F22      =ASRC3
1973 F0F22 814  =ASLC13 ASRC
1974 F0F25      =ASRC2
1975 F0F25 814  =ASLC14 ASRC
1976 F0F28      =ASRC1
1977 F0F28 814  =ASLC15 ASRC
1978 F0F2B 01           RTN
1979            ********************************************************************
1980            ********************************************************************
```

```
1981            **
1982            ** Name:       CSLCn - Shift C[W] the given # of nibbles LEFT
1983            ** Name:       CSRCn - Shift C[W] the given # of nibbles RIGHT
1984            **
1985            ** Category:   PILUTL
1986            **
1987            ** Purpose:
1988            **      Shift the C register by a given number of nibbles
1989            **
1990            ** Entry:
1991            **      None
1992            **
1993            ** Exit:
1994            **      C[W] is rotated the given # of nibbles
1995            **
1996            ** Calls:      None
1997            **
1998            ** Uses.......
1999            **  Inclusive: C[W] (rotated as per instructions)
2000            **
2001            ** Stk lvls:   0
2002            **
2003            ** NOTE: Does not alter P or carry!!!
2004            **
2005            ** History:
2006            **
2007            **    Date     Programmer              Modification
2008            **  --------   ----------    ---------------------------------
2009            **  01/03/83      NZ         Updated documentation
2010            **
2011            ********************************************************************
2012            ********************************************************************
2013 F0F2D      =CSRC8
2014 F0F2D 812  =CSLC8  CSLC
2015 F0F30      =CSRC9
2016 F0F30 812  =CSLC7  CSLC
2017 F0F33      =CSRC10
2018 F0F33 812  =CSLC6  CSLC
2019 F0F36      =CSRC11
2020 F0F36 812  =CSLC5  CSLC
2021 F0F39      =CSRC12
2022 F0F39 812  =CSLC4  CSLC
2023 F0F3C      =CSRC13
2024 F0F3C 812  =CSLC3  CSLC
2025 F0F3F      =CSRC14
2026 F0F3F 812  =CSLC2  CSLC
2027 F0F42      =CSRC15
2028 F0F42 812  =CSLC1  CSLC
2029 F0F45 01           RTN
2030            *-
2031            *-
2032 F0F47      =CSRC7
2033 F0F47 816  =CSLC9  CSRC
2034 F0F4A      =CSRC6
2035 F0F4A 816  =CSLC10 CSRC
```

```
2036 F0F4D      =CSRC5
2037 F0F4D 816  =CSLC11 CSRC
2038 F0F50      =CSRC4
2039 F0F50 816  =CSLC12 CSRC
2040 F0F53      =CSRC3
2041 F0F53 816  =CSLC13 CSRC
2042 F0F56      =CSRC2
2043 F0F56 816  =CSLC14 CSRC
2044 F0F59      =CSRC1
2045 F0F59 816  =CSLC15 CSRC
2046 F0F5C 01           RTN
2047            ****************************************************************
2048            ****************************************************************
2049            **
2050            ** Name:       BLANKC - Load C[W] with 8 blanks
2051            **
2052            ** Category:   GENUTL
2053            **
2054            ** Purpose:
2055            **      Load 8 blanks into C[W]
2056            **
2057            ** Entry:
2058            **      None
2059            **
2060            ** Exit:
2061            **      P=0, C[W]="        "
2062            **      Carry unchanged!!!
2063            **
2064            ** Calls:      None
2065            **
2066            ** Uses.......
2067            **  Inclusive: C[W],P
2068            **
2069            ** Stk lvls:   None
2070            **
2071            ** History:
2072            **
2073            **    Date     Programmer             Modification
2074            **  --------   ----------    ---------------------------------
2075            **  12/06/82      NZ         Added routine and documentation
2076            **
2077            ****************************************************************
2078            ****************************************************************
2079 F0F5E 20   =BLANKC P=     0
2080 F0F60 3F02         LCASC  \        \
           0202
           0202
           0202
           02
2081 F0F72 01           RTN
2082            ****************************************************************
2083            ****************************************************************
2084            **
2085            ** Name:       D1=AVE,D1=AVS,D1@AVE,D1@AVS - Set D1 to pointer
2086            **
```

```
2087            ** Category:   PTRUTL
2088            **
2089            ** Purpose:
2090            **      Set D1 either at AVMEME/AVMEMS or (AVMEME)/(AVMEMS)
2091            **
2092            ** Entry:
2093            **      None
2094            **
2095            ** Exit:
2096            **      D1 @ pointer, carry unchanged
2097            **      (D1@xxx:C[A]=pointer address)
2098            **
2099            ** Calls:      None
2100            **
2101            ** Uses.......
2102            **  Inclusive: C[A],D1
2103            **
2104            ** Stk lvls:   0 (D1@xxx uses 1 stack level)
2105            **
2106            ** NOTE: Does not change P or carry!
2107            **
2108            ** History:
2109            **
2110            **    Date      Programmer              Modification
2111            **  --------    ----------    ---------------------------------
2112            **  02/07/83      NZ          Changed D1=C to CD1EX (Exit cond)
2113            **  01/12/83      NZ          Added documentation
2114            **
2115            ****************************************************************
2116            ****************************************************************
2117 F0F74 1F00 =D1=AVE D1=(5) =AVMEME
           000
2118 F0F7B 01           RTN
2119            *-
2120            *-
2121 F0F7D 1F00 =D1=AVS D1=(5) =AVMEMS
           000
2122 F0F84 01           RTN
2123            *-
2124            *-
2125 F0F86 7AEF =D1@AVE GOSUB  D1=AVE
2126 F0F8A 147  =ReadD1 C=DAT1 A
2127 F0F8D 137          CD1EX                 Leave pointer address in C[A]
2128 F0F90 01           RTN
2129            *-
2130            *-
2131 F0F92 77EF =D1@AVS GOSUB  D1=AVS
2132 F0F96 63FF         GOTO   ReadD1
2133 F0F9A              END
```

```
=A-MULT  Abs  986658 #F0E22 -  1661
 ADCYCL  Abs  986691 #F0E43 -  1676  1679
=ALRNOG  Abs  986792 #F0EA8 -  1797
=ALRNOS  Abs  986842 #F0EDA -  1862
=ARLNOS  Abs  986818 #F0EC2 -  1848
=ASLC1   Abs  986897 #F0F11 -  1960
=ASLC10  Abs  986905 #F0F19 -  1967
=ASLC11  Abs  986908 #F0F1C -  1969
=ASLC12  Abs  986911 #F0F1F -  1971
=ASLC13  Abs  986914 #F0F22 -  1973
=ASLC14  Abs  986917 #F0F25 -  1975
=ASLC15  Abs  986920 #F0F28 -  1977
=ASLC2   Abs  986894 #F0F0E -  1958
=ASLC3   Abs  986891 #F0F0B -  1956
=ASLC4   Abs  986888 #F0F08 -  1954
=ASLC5   Abs  986885 #F0F05 -  1952
=ASLC6   Abs  986882 #F0F02 -  1950
=ASLC7   Abs  986879 #F0EFF -  1948
=ASLC8   Abs  986876 #F0EFC -  1946
=ASLC9   Abs  986902 #F0F16 -  1965
=ASRC1   Abs  986920 #F0F28 -  1976
=ASRC10  Abs  986882 #F0F02 -  1949
=ASRC11  Abs  986885 #F0F05 -  1951
=ASRC12  Abs  986888 #F0F08 -  1953
=ASRC13  Abs  986891 #F0F0B -  1955
=ASRC14  Abs  986894 #F0F0E -  1957
=ASRC15  Abs  986897 #F0F11 -  1959
=ASRC2   Abs  986917 #F0F25 -  1974
=ASRC3   Abs  986914 #F0F22 -  1972
=ASRC4   Abs  986911 #F0F1F -  1970   922
=ASRC5   Abs  986908 #F0F1C -  1968
=ASRC6   Abs  986905 #F0F19 -  1966
=ASRC7   Abs  986902 #F0F16 -  1964
=ASRC8   Abs  986876 #F0EFC -  1945
=ASRC9   Abs  986879 #F0EFF -  1947
=ATNCHK  Abs  986053 #F0BC5 -   982
 ATNCHc  Abs  986094 #F0BEE -  1001   983   994   996
 ATNFLG  Ext                -   989
 AVMEME  Ext                -  2117
 AVMEMS  Ext                -  2121
 Attn    Ext                -   982
=BLANKC  Abs  986974 #F0F5E -  2079
 CHKSET  Ext                -  1101
 CHKST+  Abs  986227 #F0C73 -  1133  1100
 CHKST.  Abs  986184 #F0C48 -  1107  1104
=CHKSTS  Abs  986148 #F0C24 -  1097  1091
 CHKSTe  Abs  986225 #F0C71 -  1132
 CHKSTn  Abs  986234 #F0C7A -  1138  1112
=CONVUC  Abs  986733 #F0E6D -  1742
=CSLC1   Abs  986946 #F0F42 -  2028
=CSLC10  Abs  986954 #F0F4A -  2035
=CSLC11  Abs  986957 #F0F4D -  2037
=CSLC12  Abs  986960 #F0F50 -  2039
=CSLC13  Abs  986963 #F0F53 -  2041
=CSLC14  Abs  986966 #F0F56 -  2043
```

```
=CSLC15  Abs  986969 #F0F59 -  2045
=CSLC2   Abs  986943 #F0F3F -  2026
=CSLC3   Abs  986940 #F0F3C -  2024
=CSLC4   Abs  986937 #F0F39 -  2022
=CSLC5   Abs  986934 #F0F36 -  2020
=CSLC6   Abs  986931 #F0F33 -  2018
=CSLC7   Abs  986928 #F0F30 -  2016
=CSLC8   Abs  986925 #F0F2D -  2014
=CSLC9   Abs  986951 #F0F47 -  2033
=CSRC1   Abs  986969 #F0F59 -  2044
=CSRC10  Abs  986931 #F0F33 -  2017
=CSRC11  Abs  986934 #F0F36 -  2019
=CSRC12  Abs  986937 #F0F39 -  2021
=CSRC13  Abs  986940 #F0F3C -  2023
=CSRC14  Abs  986943 #F0F3F -  2025
=CSRC15  Abs  986946 #F0F42 -  2027
=CSRC2   Abs  986966 #F0F56 -  2042
=CSRC3   Abs  986963 #F0F53 -  2040
=CSRC4   Abs  986960 #F0F50 -  2038
=CSRC5   Abs  986957 #F0F4D -  2036
=CSRC6   Abs  986954 #F0F4A -  2034
=CSRC7   Abs  986951 #F0F47 -  2032
=CSRC8   Abs  986925 #F0F2D -  2013
=CSRC9   Abs  986928 #F0F30 -  2015
=D1=AVE  Abs  986996 #F0F74 -  2117  2125
=D1=AVS  Abs  987005 #F0F7D -  2121  2131
=D1@AVE  Abs  987014 #F0F86 -  2125
=D1@AVS  Abs  987026 #F0F92 -  2131
 DDL     Ext                  937
 DDT     Ext                  901
=DTOH    Abs  986471 #F0D67 -  1476
 DTOH0   Abs  986486 #F0D76 -  1481  1485  1507
 DTOH1   Abs  986499 #F0D83 -  1488  1482
 DTOH2   Abs  986518 #F0D96 -  1496  1491
 DTOH3   Abs  986532 #F0DA4 -  1502  1497
 DTOH4   Abs  986541 #F0DAD -  1506  1493  1499  1503
 DevID   Ext                  721
 DevTyp  Ext                  660
 Device  Ext                  1044  1121
 DsLoop  Ext                  656
 DsNull  Ext                  646
=END     Abs  985207 #F0877 -  250   232
=ENDFN   Abs  985173 #F0855 -  236
=ENDST   Abs  985163 #F084B -  232
=FNDCH-  Abs  986128 #F0C10 -  1090  368
=FNDCHK  Abs  986139 #F0C1B -  1095
 FNDMB-  Ext                  1090
 FNDMBX  Ext                  1095
 FR00-0  Abs  985148 #F083C -  176   171
 FR00-1  Abs  985139 #F0833 -  170   165
 FR00-2  Abs  985130 #F082A -  164   159
 FR00-3  Abs  985121 #F0821 -  158   153
 FR00XX  Abs  985076 #F07F4 -  124   118
 FR0XXX  Abs  985065 #F07E9 -  108    80
 FR11XX  Abs  985052 #F07DC -   97    90
```

```
 FR1XXX  Abs  985047 #F07D7 -    89
=FRAME+  Abs  985026 #F07C2 -    67  1186
=FRAME-  Abs  985040 #F07D0 -    77
 FRAME0  Abs  985040 #F07D0 -    78    69
 FREND   Abs  985146 #F083A -   173    99   135   151
 FRERR   Abs  985159 #F0847 -   183   141   177
 FRERRS  Abs  985157 #F0845 -   182   144
=GADDR   Abs  985492 #F0994 -   635   492
 GADDR$  Abs  985621 #F0A15 -   714   709
 GADDR&  Abs  985747 #F0A93 -   790   931
 GADDR'  Abs  985562 #F09DA -   676   649   800
 GADDR+  Abs  985923 #F0B43 -   890   889
 GADDR-  Abs  985734 #F0A86 -   778   773
 GADDR0  Abs  985516 #F09AC -   652   641
 GADDR1  Abs  985533 #F09BD -   660   654
 GADDR2  Abs  985603 #F0A03 -   699   672
 GADDR3  Abs  985607 #F0A07 -   702   662
 GADDR4  Abs  985703 #F0A67 -   768   825
 GADDR5  Abs  985790 #F0ABE -   815   771   777   780
 GADDR6  Abs  985825 #F0AE1 -   840   724
 GADDR7  Abs  985930 #F0B4A -   892   881
 GADDR8  Abs  985948 #F0B5C -   900   894
 GADDR9  Abs  986049 #F0BC1 -   945   941
 GADDR?  Abs  985614 #F0A0E -   711   713
 GADDRN  Abs  985508 #F09A4 -   647   657
 GADDRd  Abs  985646 #F0A2E -   732   723
 GADDRe  Abs  985816 #F0AD8 -   834   503   796
 GADDRf  Abs  985811 #F0AD3 -   828   824   876
 GADDRn  Abs  985772 #F0AAC -   803   699   754   829   949
 GADDRu  Abs  985816 #F0AD8 -   833   846
 GADDRv  Abs  985869 #F0B0D -   866   942
 GADDr-  Abs  985941 #F0B55 -   895   904
 GADDrn  Abs  986019 #F0BA3 -   934   891   896   918   925
 GET     Ext                -  1312
=GETALR  Abs  986786 #F0EA2 -  1795  1802
=GETDev  Abs  986096 #F0BF0 -  1036   375
 GETERR  Ext                -   434
 GETHS2  Ext                -  1098
 GETID   Ext                -   768
 GETMBX  Ext                -   251
 GETST   Ext                -   440  1105
=GTYPE   Abs  986260 #F0C94 -  1227
 GTYPE-  Abs  986283 #F0CAB -  1232  1244
 GTYPE0  Abs  986300 #F0CBC -  1241  1240
 GTYPE1  Abs  986310 #F0CC6 -  1245  1250  1262
 GTYPE2  Abs  986314 #F0CCA -  1249  1238
 GTYPE3  Abs  986326 #F0CD6 -  1254  1261
 GTYPE4  Abs  986334 #F0CDE -  1259  1252
 Get     Abs  986386 #F0D12 -  1312  1184
 Geterr  Abs  985321 #F08E9 -   434   429   746  1103
 Getmbx  Abs  985207 #F0877 -   251   242   456   481  1124
=HTOD    Abs  986556 #F0DBC -  1548
 HTOD1   Abs  986566 #F0DC6 -  1556  1559
 HTOD2   Abs  986577 #F0DD1 -  1562  1557
 HTOD3   Abs  986584 #F0DD8 -  1564  1567
```

```
=HTODX   Abs  986596 #F0DE4 -  1600
 HTODX1  Abs  986609 #F0DF1 -  1605  1608  1627
 HTODX2  Abs  986621 #F0DFD -  1611  1606
 HTODX3  Abs  986633 #F0E09 -  1621  1613
 HTODX4  Abs  986653 #F0E1D -  1628  1622
 HTODXr  Abs  986629 #F0E05 -  1617  1629
=LISTEN  Abs  986353 #F0CF1 -  1299  1349
 LOOPST  Ext                  1038  1118
 Loop    Ext                   386   652
 M-STRT  Abs  986685 #F0E3D -  1673  1668
=MTYL    Abs  986392 #F0D18 -  1347   934
=MTYLC   Abs  986406 #F0D26 -  1351
=MTYLL   Abs  986399 #F0D1F -  1349
 NXTDGT  Abs  986680 #F0E38 -  1671  1674  1681
 Null    Ext                   383   639
 OVFLOW  Abs  986714 #F0E5A -  1686  1677
 PRMSG1  Abs  986449 #F0D51 -  1435  1441
=PRMSGA  Abs  986446 #F0D4E -  1434
=PUTALR  Abs  986834 #F0ED2 -  1856  1866
=PUTARL  Abs  986810 #F0EBA -  1842  1852
 PUTC    Ext                  1301
 PUTC+   Ext                   740
 PUTC=D  Abs  986359 #F0CF7 -  1300  1392
 PUTD    Ext                  1309
=PUTDX   Abs  986858 #F0EEA -  1904  1909
 PUTE    Ext                   424  1231
=PUTGF   Abs  986246 #F0C86 -  1183  1232
=PUTGF+  Abs  986242 #F0C82 -  1182   669   873
=PUTGF-  Abs  986238 #F0C7E -  1180   416   483   793   821
 Putc    Abs  986362 #F0CFA -  1301  1182  1306  1352  1389
=Putd    Abs  986380 #F0D0C -  1309  1437  1848  1862  1904
=RANGE   Abs  986771 #F0E93 -  1755  1743  1751
=RANGEA  Abs  986755 #F0E83 -  1750  1737
=RANGEN  Abs  986765 #F0E8D -  1754
 READRG  Ext                   906
 RESTRT  Ext                   455   460
=ReadD1  Abs  987018 #F0F8A -  2126  2132
 Rewind  Ext                   936
 SEEKA   Ext                   893
 SETLP   Ext                   357
 SFLAG?  Ext                   499
=START   Abs  985213 #F087D -   353
 START!  Abs  985327 #F08EF -   440   430
 START#  Abs  985295 #F08CF -   423   414
=START+  Abs  985219 #F0883 -   359
=START-  Abs  985222 #F0886 -   363
 START0  Abs  985336 #F08F8 -   442
 START1  Abs  985348 #F0904 -   450   449
 START2  Abs  985377 #F0921 -   460   443   454
 START3  Abs  985453 #F096D -   487   401   451   457
 START5  Abs  985488 #F0990 -   502   420   486
 STARTS  Abs  985312 #F08E0 -   429   417   419
 STARTd  Abs  985266 #F08B2 -   397   382   385   388
 STARTn  Abs  985270 #F08B6 -   407   376
 STARTp  Abs  985424 #F0950 -   476
```

```
 STARTs  Abs  985433 #F0959 -   480   468   477
 STARtE  Abs  985319 #F08E7 -   431
 SWAP01  Ext                -   469   472
=TALK    Abs  986436 #F0D44 -  1391
 TSTAT   Ext                -   880
 TSTATA  Ext                -   903
=UCRANG  Abs  986726 #F0E66 -  1737
=ULYL    Abs  986346 #F0CEA -  1297
=UNLPUT  Abs  986368 #F0D00 -  1304   245   732  1297  1347  1386
 UNT     Ext                -   243   807
=UTLEND  Abs  985185 #F0861 -   242   237
 VolLbl  Ext                -   844
=YTML    Abs  986416 #F0D30 -  1386  1227
=YTMLL   Abs  986423 #F0D37 -  1388
 bPILAI  Ext                -   470
 eABORT  Ext                -   997
 eBADMD  Ext                -   392  1133
 eNEWTA  Ext                -   888
 eNOFND  Ext                -   810
 eNORDY  Ext                -  1260
 ePIL    Ext                -   393   811   836  1134  1255
 eRANGE  Ext                -  1628
 eTAPE   Ext                -   885   895
 eUNEXP  Ext                -   835  1253
 fIEXTD  Ext                -   462
 fINZ4   Ext                -   452
 i/OFND  Ext                -   471
 mADDRL  Ext                -  1299
 mADDRM  Ext                -  1351  1388
 mADDRT  Ext                -  1391
 mAUTOA  Ext                -   467   476
 mFIND1  Ext                -   668
 mFINDD  Ext                -   867
 mGETCA  Ext                -   792
 mINCCA  Ext                -   820
 mPULOP  Ext                -   415
 mRSTCA  Ext                -   739
 mSAI    Ext                -  1230
 mSDA    Ext                -   905
 mTAKEI  Ext                -   423
 mUNL    Ext                -  1305
 nXTSTM  Ext                -   233
 p3DATA  Ext                -    73
 pACK    Ext                -   150
 pADDR   Ext                -   119   485   671   795   823   875
 pDATA   Ext                -   104  1237
 pEOT    Ext                -   160  1249
 pETE    Ext                -   154
 pHALTD  Ext                -   166
 pIFC    Ext                -   172
 pSTATE  Ext                -   418   828  1251
 pTERM   Ext                -   178
 pUTYPE  Ext                -   183
 sCONTR  Ext                -  1111
=sFLAG?  Abs  985481 #F0989 -   499
```

```
 sMANUL  Ext                  -  1099
 sReadd  Ext                  -   359   413   442   461
 sUNCNF  Ext                  -   448
 sflag?  Abs  985474 #F0982   -   496   453   463
```

Input Parameters

  Source file name is NZ&GPR::MS

  Listing file name is NZ/GPR:TI:ML::-1

  Object file name is NZ%GPR:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
1            *
2            *       N   N  ZZZZZ   &     BBBB     A     SSS
3            *       N   N      Z  & &    B   B   A A   S   S
4            *       NN  N     Z   & &    B   B  A   A  S
5            *       N N N    Z     &     BBBB   A   A   SSS
6            *       N  NN   Z     & & &  B   B  AAAAA      S
7            *       N   N  Z      &  &   B   B  A   A  S   S
8            *       N   N  ZZZZZ   && &  BBBB   A   A   SSS
9            *
10           *
11                   TITLE  BASIC ROUTINES <840116.1657>
12 F0F9A             ABS    #F0F9A        TI%HP6 address (fixed)
13           ****************************************************************
14           ****************************************************************
15           **
16           ** Name:        PRTIS - Poll handler for the PRINT statement
17           ** Name:        PRTIS+ - Poll handler for pPRTCL (D1 @ address)
18           ** Name:        PRTISc - Address device as listener (D1 @ addr)
19           **
20           ** Category:    POLL
21           **
22           ** Purpose:
23           **      Handle pPRTIS/pPRTCL/... (address device as listener,
24           **      me as talker, load address of routine to send data)
25           **
26           ** Entry:
27           **      P=0, HEXMODE
28           **      PRTIS+,PRTISc:
29           **             D1 points to the 7 nib device assignment
30           **             FUNCD1 contains the value to return in D1
31           **
32           ** Exit:
33           **      Carry clear
34           **      If XM=0, A[A] is the address of the PRINT handler
35           **      If XM=1, Did NOT handle the poll
36           **      PRTIS+,PRTICc: D1 restored from FUNCD1
37           **
38           ** Calls:       TSAVD1,CHKASN,TRESD1,START,ULYL,MTYL
39           **
40           ** Uses.......
41           **  Inclusive: A,B,C,D[15:13,5:0],D0,P,FUNCD0[2:0],FUNCD1
42           **
43           ** Stk lvls:    4 (START)
44           **
45           ** NOTE: Does not alter D1 or status bits
46           **
47           ** History:
48           **
49           **    Date      Programmer              Modification
50           **  --------    ----------    ---------------------------------
51           **  11/29/83       NZ         Updated documentation
52           **  07/21/83       NZ         Removed check for mass storage
53           **                            device (not correct as it is)
54           **  06/23/83       NZ         Changed call to CHKMSD to inline
55           **                            code (only reference to CHKMSD)
```

```
                **  02/23/83      JH          Added A[S] flag for MeTalk status
                **  02/17/83      NZ          Removed multiple devices
                **  02/03/83      NZ          Changed MeTalk from 4 to 9 (START
                **                              destroys ST4)
                **  01/20/83      JH          Added MeTalk status, send MTA
                **  12/15/82      NZ          Updated documentation
                **
                ********************************************************************
                ********************************************************************
                SaveIt  EQU   6               Need to save this one after start
                MeTalk  EQU   9               Address me as talker
                *
F0F9A AC0       =PRTISc A=0   S               Entry for CLEAR, clear A[S] so My
F0F9D 6610              GOTO  PRTISe          Talk Adr is not sent out
                *-
                *-
F0FA1 8E00      =PRTIS  GOSUBL =TSAVD1        Save D1 in FUNCD1
      00
F0FA7 1F00              D1=(5) =IS-PRT
      000
F0FAE AC0       =PRTIS+ A=0   S               Set status to address me to talk
F0FB1 A4C               A=A-1 S               A[S]=F
F0FB4 15F6      PRTISe  C=DAT1 7
F0FB8 DA                A=C   A               Save low 3 nibs in A[A]
F0FBA 8E00              GOSUBL =CHKASN
      00
F0FC0 5F3               GONC  PRTIS2          This is assigned...do it
                *
                * If carry, check if this is "NULL" or "LOOP"
                *
F0FC3 96C               ?A#0  B
F0FC6 03                GOYES PRTIS1          If A[B]<>0, NOT "NULL"...exit
                *
                * A[B]=0...either "NULL" or "LOOP"
                *
F0FC8 B24               A=A+1 XS              Check if "NULL"
F0FCB 543               GONC  PRTIS2          If no carry, this is "LOOP"
                *
                * This is "NULL"
                *
F0FCE 7700              GOSUB PRTIS-          Get my address
F0FD2 5000              REL(5) =PREXT         (Address of part 3 handler)
      0
                *
                * Following is the part 2&3 handler for "NULL" (Doesn't use
                * anything, just clears carry)
                *
F0FD7 03        =PREXT  RTNCC
                *-
                *-
F0FD9 07        PRTIS-  C=RSTK                Pop my address back
F0FDB 137               CD1EX
F0FDE 174               D1=D1+ 5              Skip the REL(5)
F0FE1 133               AD1EX                 Leave address in A[A]
                *
```

```
107                 * Carry is CLEAR from the D1=D1+ 5 above...TRESD1 doesn't
108                 * affect the carry
109                 *
110 F0FE4 8C00       Tresd1  GOLONG =TRESD1       Restore D1, return "handled"
            00
111                 *-
112                 *-
113                 *
114                 * Not assigned or error...return, carry clear, XM=1
115                 *
116 F0FEA 1B00       PRTIS0  D0=(5) =FUNCD0
            000
117 F0FF1 146               C=DAT0 A
118 F0FF4 0A                ST=C                 Restore status bits from FUNCD0
119 F0FF6 7AEF      PRTIS1  GOSUB  Tresd1        Restore D1 from FUNCD1
120 F0FFA 21                P=     1
121 F0FFC 0D                P=P-1                Clear carry, P=0
122 F0FFE 00                RTNSXM               Return, not handled
123                 *-
124                 *-
125 F1000 1B00       PRTIS2  D0=(5) =FUNCD0       Save status bits in FUNCD0
            000
126 F1007 0B                CSTEX
127 F1009 15C2              DAT0=C 3
128 F100D 0B                CSTEX
129 F100F 846               ST=0   SaveIt        Initially say don't save it
130 F1012 859               ST=1   MeTalk        Set up MeTalk status bit...
131 F1015 B44               A=A+1  S             ...MeTalk = 1 if A[S]=F
132 F1018 450               GOC    PRTIS,        ...MeTalk = 0 if A[S]=0
133 F101B 849               ST=0   MeTalk
134 F101E D7    PRTIS,      D=C    A             Put device specifier in D[A]
135 F1020 94A               ?C=0   S             Did CHKASN say to find it?
136 F1023 50                GOYES  PRTIS"        No...don't need to save it
137 F1025 856               ST=1   SaveIt        Yes...need to save address
138 F1028 7000  PRTIS"      GOSUB  =STRT          Set up the device
139 F102C 4DB               GOC    PRTIS0        Error...can't handle the poll
140                 *
141                 * Now address listener, make me talker (conditionally)
142                 *
143 F102F 96B               ?D=0   B             Is this "LOOP"?
144 F1032 61                GOYES  PRTS01        Yes...don't change addressing
145 F1034 879               ?ST=1  MeTalk        Should I be addressed as talker?
146 F1037 A0                GOYES  PRTIS@        Yes...set it up
147 F1039 7000              GOSUB  =ULYL          No...send UNL, LAD n
148 F103D 6700              GOTO   PRTS00        (Check errors at PRTS00)
149                 *-
150                 *-
151 F1041 7616  PRTIS@      GOSUB  Mtyl          Address device as listener
152 F1045 44A   PRTS00      GOC    PRTIS0        HPIL error...don't handle it
153 F1048 866   PRTS01      ?ST=0  SaveIt        Do I need to write it out?
154 F104B C0                GOYES  PRTIS4        No...continue
155 F104D ABB               C=D    X             Yes...copy address from D[X]
156 F1050 15D2              DAT1=C 3             Write out the device address @ D1
157                 *
158                 * Following statement is a NOP...can be removed to save 3 nibs
```

```
                * (It is a relic from old code)
                *
F1054 520       PRTIS4  GONC    PRTIS4          Go always
F1057 1B00      PRTIS4  DO=(5)  =FUNCDO
      000
F105E 146               C=DATO  A
F1061 0A                ST=C                    Restore caller's STatus bits
F1063 7D7F              GOSUB   Tresd1          Restore caller's D1
                *
F1067 7000              GOSUB   PRTIS5          Get my current address...
F106B 07        PRTIS5  C=RSTK                  ...pop it off...
F106D DA                A=C     A               ...move it to A[A]...
F106F 3441              LC(5)   (PRASCI)-(PRTIS5)  ...Offset of part 2 routine
      000
F1076 CA                A=A+C   A               (Address of part 2 routine in A)
F1078 03                RTNCC                   Done, handled
                ********************************************************************
                ********************************************************************
                **
                ** Name:        PRASCI - Send ASCII characters to the loop
                **
                ** Category:    PILI/O
                **
                ** Purpose:
                **      Send the ASCII characters to the loop (already set up)
                **
                ** Entry:
                **      MBOX^ points to the desired mailbox
                **      A[A] contains the length of the string in bytes
                **      D[A] is the start address of the string
                **
                ** Exit:
                **      If loop error, jumps to ERRORX
                **      P=0
                **      D1 positioned following last character sent
                **
                ** Calls:       GETMBX,WRITIT,TSAVDO,TRESDO,<ERRORX>
                **
                ** Uses.......
                **  Inclusive: A[A],C,D1,P,FUNCDO,ST[8,3:0]
                **
                ** Stk lvls:    3 (pushed DO;WRITIT)(pushed DO;TRESDO)
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------    ---------------------------------
                **  12/15/82        NZ        Updated documentation
                **  01/27/83        NZ        Modified entry, exit save method,
                **                            added exit condition on D1
                **
                ********************************************************************
                ********************************************************************
F107A D300              REL(5)  =PREND          Address of the final part
      0
```

```
F107F 09    =PRASCI C=ST
F1081 136           CDOEX                 ST into D0, D0 value into C[A]
F1084 7116          GOSUB  TsavD0         Save status in FUNCD0
F1088 06            RSTK=C                Save D0 on RSTK
F108A 8E00          GOSUBL =GETMBX        Get the mailbox address
      00
F1090 DB            C=D    A
F1092 135           D1=C                  Set D1 to the start of the buffer
                *
                * Now D1-->buffer, A[A] is length in bytes, D0-->mailbox
                * Loop is addressed (Talker and Listener(s))
                *
F1095 840           ST=0   =LoopOK        Do not abort with one ATTN hit
F1098 8E00          GOSUBL =WRITIT        Transfer the data to the loop
      00
F109E 4F0           GOC    PRASER         Error if carry set
F10A1 7AF5 PRASEX   GOSUB  TresD0         Get status back to D0
F10A5 07            C=RSTK                Get old D0 from RSTK
F10A7 136           CDOEX                 Now D0 restored, ST in C[X]
F10AA 0A            ST=C                  Restore the status bits
F10AC 01            RTN
                *-
                *-
F10AE 890   PRASER  ?P=    0              Is this just an interrupt?
F10B1 0F            GOYES  PRASEX         Yes...continue
                *
                * No need to pop RSTK...jumping directly to BSERR
                *
F10B3 6155          GOTO   ErrorX         Error...jump to ERRORX --> BSERR
                ****************************************************************
                ****************************************************************
                **
                ** Name:       PREND - Clean up the loop after PRINT/OUTPUT
                **
                ** Category:   LOCAL
                **
                ** Purpose:
                **      Clean up the loop after a PRINT/OUTPUT sequence
                **
                ** Entry:
                **      Device(s) are addressed as listener(s)
                **      MBOX^ points to the mailbox used
                **
                ** Exit:
                **      D0 points to the mailbox used
                **      Carry clear (P may be non-zero)
                **
                ** Calls:      D1=SRO,SAVEIT,UTLEND
                **
                ** Uses.......
                **  Inclusive: A,B,C,D,R2,R3,D0,D1,P,ST[3:0]
                **
                ** Stk lvls:   4 (UTLEND)(SAVEIT)
                **
                ** History:
```

```
                **
                **    Date     Programmer             Modification
                **  --------   ----------   --------------------------------
                **  11/29/83      NZ        Updated documentation
                **  12/15/82      NZ        Added documentation
                **
                ****************************************************************
                ****************************************************************
F10B7           =PREND
                *
                * If device code equals OUTPTt, then need to deallocate the
                * buffer!
                *
F10B7 7CC5              GOSUB  D1=SRO        Device code
F10BB 14F               C=DAT1 B             Read in 1 nib
F10BE 80D0              P=C    0             Copy device code to P
F10C2 1D00              D1=(2) (=STMTR1)+2   Point to device spec
F10C6 14F               C=DAT1 B
F10C9 96A               ?C=0   B             NULL or LOOP?
F10CC 41                GOYES  PREND         Yes...exit cleanly
                *
F10CE 880               ?P#    =OUTPTt
F10D1 90                GOYES  PREND1
F10D3 AF2               C=0    W
F10D6 7785              GOSUB  Saveit        (This will deallocate the buffer)
F10DA           PREND1
                *
                * Unaddress all talkers and listeners
                *
F10DA 8E00              GOSUBL =UTLEND
      00
F10E0 03        PRENDE  RTNCC                Exit with carry clear
                ****************************************************************
                ****************************************************************
                **
                ** Name:       OUTPUT - Execute the OUTPUT statement
                **
                ** Category:   STEXEC
                **
                ** Purpose:
                **      Send output to the specified device(s)
                **
                ** Entry:
                **      DO at tokenized device specifier
                **
                ** Exit:
                **      Through mainframe PRINT*
                **
                ** Calls:      GETDID,SAVEIT,TRESDO,<PRINT*>,<ERRORX>
                **
                ** Uses.......
                **  Inclusive: A,B,C,D,R0-R4,D0,D1,P,FUNCxx,STMTD1[3:0],STMTR1,
                **             ST[11:0],all RAM that EXPEXC is permitted to use
                **
                ** Stk lvls:   7 (GETDID)
```

```
318                **
319                ** History:
320                **
321                **    Date     Programmer          Modification
322                **   --------  ----------   --------------------------------
323                **  11/29/83      NZ        Updated documentation
324                **  03/15/83      NZ        Replaced GETMUL with GETDID
325                **  12/15/82      NZ        Wrote code and documentation
326                **
327                ****************************************************************
328                ****************************************************************
329 F10E2 0000          REL(5) =OUTPd          OUTPUT decompile
          0
330 F10E7 0000          REL(5) =OUTPp          OUTPUT parse
          0
331 F10EC 8E00 =OUTPUT GOSUBL =GETDID         Get device specifier
          00
332 F10F2 414           GOC    OUTPer          Error with device or loop
333 F10F5 1F00          D1=(5) (=STMTR1)+2     (This is where I save the 7 nibs)
          000
334 F10FC AF0           A=0    W               Clear position, length
335 F10FF 159A          DAT1=A 11              (STMTR1)+9 is position, width
336 F1103 7A55          GOSUB  Saveit          Save the source @ D1
337 F1107 7495          GOSUB  TresdO          Restore the PC (saved by GETDID)
338 F110B 1F00          D1=(5) =EOLLEN         Point to EOL length, EOL string
          000
339 F1112 15F6          C=DAT1 7               Read EOLLEN, EOL string
340 F1116 1E00          D1=(4) (=STMTRO)+11    Position to CKINFO location
          00
341 F111C 15D6          DAT1=C 7               Write it out EOL info out
342 F1120 1CB           D1=D1- 12              Position to MLFFLG
343                *****
344                *
345                *       LC(2)  (=OUTPTt)*16+#F Set MLFFLG="F", type=OUTPTt
346 F1123 31F           NIBHEX 31F
347 F1126 0             CON(1) =OUTPTt
348                *
349                *****
350 F1127 14D           DAT1=C B               Write the info out to MLFFLG
351                *
352                * Now have written the info needed for the hPRTCL handler to
353                * do its job
354                *
355 F112A 161           DO=DO+ 2               Skip the t@ used to stop GETDID
356 F112D 8D00          GOVLNG =PRINT*         Now continue with PRINT handler
          000
357                *-
358                *-
359 F1134 60D4 OUTPer  GOTO   Errorx
360                ****************************************************************
361                ****************************************************************
362                **
363                ** Name:        PRNTIS - Reassign HPIL PRINT device
364                ** Name:        DISPIS - Reassign HPIL DISPLAY device
365                **
```

```
366                 ** Category:   STEXEC
367                 **
368                 ** Purpose:
369                 **      PRNTIS executes the PRINTER IS statement, and DISPIS
370                 **      executes the DISPLAY IS statement.
371                 **
372                 ** Entry:
373                 **      D0 points to the device specifier
374                 **
375                 ** Exit:
376                 **      Exits through ENDST if no error, ERRORX if error
377                 **
378                 ** Calls:      D1=DST,SAVED0,GETDID,RESTD0,SWAP01,SAVEIT,
379                 **             D1=DSX,PILCNF,<ENDST>,<ERRORX>
380                 **
381                 ** Uses.......
382                 **  Inclusive: A,B,C,D,R0-R4,D0,D1,P,FUNCxx,STMTD0,STMTD1,
383                 **             ST[11:0],all RAM that EXPEXC is permitted to use
384                 **
385                 ** Stk lvls:   7 (GETDID)
386                 **
387                 ** History:
388                 **
389                 **    Date     Programmer          Modification
390                 **  --------   ----------   --------------------------------
391                 **  01/06/84      NZ        Changed order of DISPIS to set up
392                 **                          to search AFTER calling GETDID
393                 **  11/29/83      NZ        Updated documentation and added
394                 **                          PRNT00 as an external entry point
395                 **  05/17/83      NZ        Corrected mod of 5/4/83 to error
396                 **                          for bad device spec
397                 **  05/04/83      NZ        Modified return from GETDID to
398                 **                          match new exit conditions of same
399                 **  03/18/83      NZ        Used STMTD0 instead of STMTD1 to
400                 **                          save address through GETDID
401                 **  02/18/83      NZ        Added call to PILCNF for DISPIS
402                 **  12/15/82      NZ        Updated documentation
403                 **
404                 ****************************************************************
405                 ****************************************************************
406 F1138 0000 0       REL(5) =PRNTSd      "PRINTER IS" DECOMPILE
407 F113D 0000 0       REL(5) =PRNTSp      "PRINTER IS" PARSE
408 F1142         =DISPIS
409 F1142 3400 000     LC(5)  =IS-DSP
410                 *
411                 * Following statement is a "Go always" because the LEX table
412                 * entry for DISPIS is earlier in memory than DISPIS, hence
413                 * the calculation of the execution address leaves carry clear
414                 *
415 F1149 512           GONC   PRNT00       Go always
416                 *-
417                 *-
```

```
418 F114C 15D0 DISPI+  DAT1=C 1            Write out the bits
419 F1150 8E00         GOSUBL =PILCNF      Set up DSPCNX if needed
          00
420 F1156 69E4 PRNT50  GOTO   Endst        Clean up, goto next statement
421            *-
422            *-
423 F115A 0000         REL(5) =PRNTSd      "PRINTER IS" decompile
          0
424 F115F 0000         REL(5) =PRNTSp      "PRINTER IS" parse
          0
425 F1164 3400 =PRNTIS LC(5)  =IS-PRT
          000
426 F116B 136  =PRNTOO CDOEX               Save PC in C[A],put address in DO
427 F116E 8E00         GOSUBL =SAVEDO      Save location in STMTDO
          00
428 F1174 136          CDOEX               Restore PC from C[A]
429 F1177 8E00         GOSUBL =GETDID      Get device specifier
          00
430            *
431            * Following two routines do not change carry
432            *
433 F117D 8E00         GOSUBL =RESTDO      Now DO @ intended location
          00
434 F1183 8E00         GOSUBL =SWAPO1      Swap DO, D1
          00
435            *
436            * Now D1 is at the destination
437            *
438 F1189 551          GONC   PRNT45       No error...save it in RAM
439            *
440            * Check for *, "" (Address=0, carry set)
441            *
442 F118C 8AF          ?D#0   A
443 F118F A5           GOYES  PRNTER       Not a valid device spec
444 F1191 880          ?P#    =eDSPEC      Is it "", *, or "*"?
445 F1194 55           GOYES  PRNTER       No...error
446            *
447            * Device is "*"...undo it
448            *
449 F1196 AF2          C=0    W
450 F1199 A7E          C=C-1  W
451 F119C AC2          C=0    S            Indicate "fits" in 7 nibs
452 F119F 7EB4 PRNT45  GOSUB  Saveit       Save source @ D1
453            *
454            * Check if this is DISPLAY IS
455            *
456 F11A3 133          AD1EX
457 F11A6 3400         LC(5)  =IS-DSP
          000
458 F11AD 8A6          ?A#C   A            Is it DISPLAY?
459 F11B0 6A           GOYES  PRNT50       No...exit
460 F11B2 8E00         GOSUBL =D1=DSX      Yes...point to DSPCHX (address)
          00
461 F11B8 D2           C=0    A
462 F11BA 145          DAT1=C A            Clear DISCHX for case of "*"
```

```
463 F11BD 8E00          GOSUBL =D1=DST       Point to DSPSET
          00
464 F11C3 307           LC(1)  7             Printr,Wallby,LoopOK=1; DispOK=0
465 F11C6 658F          GOTO   DISPI+        Go always (reset DSPCHX, clean up)
466             *-
467             *-
468 F11CA 0             CON(1) =FIXSPC       1 nibble available here
469 F11CB               BSS    1-1
470             ****************************************************************
471             ****************************************************************
472             **
473             ** Name:       PACKD - Pack the directory of a mass storage dev
474             **
475             ** Category:   STEXEC
476             **
477             ** Purpose:
478             **      Pack a mass storage device directory
479             **
480             ** Entry:
481             **      D0 points to the device specifier
482             **
483             ** Exit:
484             **      Through NXTSTM or ERRORX
485             **
486             ** Calls:      PDIR,ENDTAP,<NXTSTM>,<ERRORX>
487             **
488             ** Uses.......
489             **  Inclusive: All CPU registers, all RAM EXPEXC is permitted
490             **             to use, STMTD0[3:0],STMTR1
491             **
492             ** Stk lvls:   7 (PDIR)
493             **
494             ** History:
495             **
496             **    Date     Programmer             Modification
497             **  --------   ----------   --------------------------------
498             **  12/21/83      NZ        Moved call to GETDID to PACKD to
499             **                          fix a stack level problem (PDIR)
500             **  11/29/83      NZ        Updated documentation
501             **
502             ****************************************************************
503             ****************************************************************
504 F11CB 0000             REL(5) =PACKd        PACK decompile
          0
505 F11D0 0000             REL(5) =PACKp        PACK parse
          0
506 F11D5 8E00 =PACKD      GOSUBL =GETDID       Get the device specifer
          00
507 F11DB 7E00             GOSUB  PDIR          Pack the directory
508 F11DF 490              GOC    PRNTER        Error during pack
509 F11E2 6302             GOTO   PACK90        ENDTAP, NXTSTM
510             *-
511             *-
512 F11E6 0             CON(1) =FIXSPC       3 nibbles available here
513 F11E7               BSS    3-1
```

```
                *-
                *-
                *
                * Error detected
                *
F11E9 6B14 PRNTER  GOTO   Errorx        If error, don't change IS-xxx
                ****************************************************************
                ****************************************************************
                **
                ** Name:       PDIR - Pack a directory (assembly language call)
                **
                ** Category:   LOCAL
                **
                ** Purpose:
                **      Pack a mass storage device directory
                **
                ** Entry:
                **      Exit conditions from GETDID
                **
                ** Exit:
                **      Carry clear: (successful pack)
                **        P=0
                **        DO points to the HPIL mailbox
                **        D[X] is the address of the mass storage device
                **        R0 is the information returned in B[W] from GDIRST
                **        R1 is the information returned in D[W] from GDIRST
                **      Carry set: (error occurred)
                **        P,C[0] are the error code
                **
                ** Calls:      CHKMAS,GDIRST,GETDR",CSRC4,NXTENT,CSRC5,CSLC5,
                **             PDIRBF,CSLC4,PBF->C,GETDR+,F->SCR,CSLC3,
                **      PBF->C:SEEKA,DDT,ULYL,DDL,TSTAT,<DDT>
                **      -----
                **      PDIRBF:MTYL,DDL,CSLC4,PUTD,<PUTDR">
                **
                ** Uses.......
                **  Inclusive: A-D,R0-R4,D0,D1,P,ST[11:0]
                **
                ** Stk lvls:   4 (GDIRST)
                **
                ** PDIR:Set up the loop (START)
                **      Check for mass storage device
                **      Get directory information (GDIRST)
                **      (PTRC is current directory entry)
                **       (PTRC is B[3:0])
                **      (PTRD is where next non-purged directory entry goes)
                **       (PTRD is B[15:12])
                **    1:Seek correct record & read directory entry
                **    2:IF (physical end of directory) THEN GOTO 8..:
                **      IF (logical end of directory) THEN GOTO 8:
                **      Increment PTRC
                **      IF (PTRC crossed record boundary) THEN
                **         Decrement record count (D[8:5])
                **      IF (entry is purged) THEN GOTO 3:
                **      Write entry at PTRD (Buffer 1)
```

```
569                **      Increment PTRD
570                **      IF (PTRD not at start of record) THEN GOTO 3:
571                **      Write out buffer 1 contents to tape
572                **      GOTO 1:
573                **      --
574                **    3:Read directory entry
575                **      GOTO 2:
576                **      --
577                **    8:Write out EOD marker (if not at physical EOD)
578                **    9:RETURN
579                **
580                ** History:
581                **
582                **    Date      Programmer            Modification
583                **  --------    ----------    --------------------------------
584                **  12/21/83       NZ         Removed call to GETDID to fix a
585                **                            bug (stack levels)
586                **  05/25/83       NZ         Added mass storage check in PDIR
587                **  01/06/83       NZ         Rewrote algorithm, documented it
588                **  12/15/82       NZ         Updated documentation
589                **
590                ********************************************************************
591                ********************************************************************
592 F11ED 400    =PDIR    RTNC                  Error with device specifier
593 F11F0 8E00           GOSUBL =CHKMAS         Check for mass storage
           00
594 F11F6 400            RTNC                  Not mass storage...error
595 F11F9 23             P=     3
596 F11FB 304            LC(1)  4              This is Acc ID=16 (for MOVEFL)
597 F11FE A87            D=C    P
598 F1201 8E00           GOSUBL =GDIRST         Get the directory start info
           00
599 F1207 400            RTNC
600 F120A AF9            C=B    W
601 F120D 108            R0=C                  Save B[W] in R0 for PACK
602 F1210 AFB            C=D    W
603 F1213 109            R1=C                  Save D[W] in R1 for PACK
604 F1216 8E00 PDIR10    GOSUBL =GETDR"        Get the entry from B[3:0]
           00
605 F121C 400            RTNC                  Error
606 F121F 90D  PDIR20    ?B#0   P              New record?
607 F1222 31             GOYES  PDIR22         No...continue
608                *
609                * New record...check for end of directory
610                *
611                PhyEOD   EQU    0
612 F1224 850            ST=1   PhyEOD         Physical End Of Directory
613 F1227 AFB            C=D    W
614 F122A 7F44           GOSUB  Csrc4
615 F122E F6             CSR    A              Now C[3:0] is count, C[4]=0
616 F1230 8AA            ?C=0   A              Is the record count zero?
617 F1233 A7             GOYES  PDIR90         Yes...physical EOD
618 F1235 840  PDIR22    ST=0   PhyEOD         No...not physical EOD.
619 F1238 173            D1=D1+ 4              Move to TYPE
620 F123B 15F3           C=DAT1 4              Read in file type
```

```
                        *
                        * Check for end of directory (FFFF)
                        *
F123F E6                C=C+1  A
F1241 F2                CSL    A              Now C[4:1] is type+1, C[0]=0
F1243 8AA               ?C=0   A              Is this logical EOD?
F1246 76                GOYES  PDIR90         Yes...done
F1248 DD                BCEX   A              No
F124A 8E00              GOSUBL =NXTENT        Increment directory pointer
      00
F1250 DD                BCEX   A              (Carry if new record)
F1252 521               GONC   PDIR24         Not new record...continue
                        *
                        * New record...need to decrement record count in D[8:5]
                        *
F1255 AFF               CDEX   W
F1258 7E14              GOSUB  Csrc5
F125C CE                C=C-1  A              C[3:0] is always >0...use C[A]
F125E 7114              GOSUB  Cslc5
F1262 AFF               CDEX   W              Replace the count, restore C[A]
F1265 F6   PDIR24       CSR    A              Type + 1 in C[3:0], C[4]=0
F1267 CE                C=C-1  A              Type in C[A]
F1269 8AA               ?C=0   A              Is this a purged entry?
F126C F2                GOYES  PDIR30         Yes...read next entry @ PTRC
                        *
                        * Non-purged entry...put directory entry into buffer 1 of tape
                        *
F126E 7190              GOSUB  PDIRBF         Write (SCRTCH) to B[12]th entry
F1272 400               RTNC
F1275 AF9               C=B    W
F1278 7AF3              GOSUB  Cslc4          Get the address of buffer 1
F127C D5                B=C    A              Save pointer in B[A] for now
F127E 8E00              GOSUBL =NXTENT
      00
F1284 75F3              GOSUB  Csrc4          Rotate back to C[15:12]
F1288 AFD               BCEX   W              Now C[A] is entry, B is restored
F128B 5F0               GONC   PDIR30         Not a new record...continue
                        *
                        * This is a new record...write buffer 1 @ recprd C[3:1]
                        *
F128E F6                CSR    A              Record # in C[X] now
F1290 7240              GOSUB  PBF->C         Write buffer 1 to @C[X]
F1294 400               RTNC                  Error
F1297 6E7F              GOTO   PDIR10         No error...go reread the record
                        *-
                        *-
                        *
                        * Wrote a new entry into buffer 1, but didn't fill buffer 1
                        *
F129B D4   PDIR30       A=B    A
F129D 814               ASRC                  A[S] is byte pointer div 32
F12A0 8E00              GOSUBL =GETDR+        Get next directory entry
      00
F12A6 400               RTNC                  Error
F12A9 657F              GOTO   PDIR20         No error...process the entry
```

```
673                *-
674                *-
675                *
676                * Reached end of directory...check whether physical or logical
677                *
678 F12AD 860  PDIR90  ?ST=0  PhyEOD        Physical EOD?
679 F12B0 21          GOYES  PDIR92        No...continue
680 F12B2 AF9         C=B    W             Yes...check if room for a new EOD
681 F12B5 7DB3        GOSUB  Cslc4         Get PTRD into C[3:0]
682 F12B9 E9          C=C-B  A             Now C[3:0] is PTRC-PTRD
683 F12BB F2          CSL    A             Now C[A]=0 iff PTRC=PTRD
684 F12BD 8AA         ?C=0   A             Is there space for an EOD mark?
685 F12C0 F0          GOYES  PDIR95        No...exit
686 F12C2       PDIR92
687                *
688                * Write an end-of-directory mark in buffer 1
689                *
690 F12C2 8E00        GOSUBL =F->SCR       Put "FFF"s in SCRTCH[63:0]
          00
691 F12C8 7730        GOSUB  PDIRBF        Put SCRTCH @ PTRD
692 F12CC 400         RTNC                 Error
693 F12CF AF9  PDIR95  C=B    W
694 F12D2 7000        GOSUB  =CSLC3        C[X] is PTRD record # now
695                *
696                * Fall into PBF->C
697                *
698                * PBF->C writes the record in buffer 1 at the record number
699                * in C[X] on the mass storage device
700                *
701 F12D6 D0   PBF->C  A=0    A
702 F12D8 ABA         A=C    X
703 F12DB 8E00        GOSUBL =SEEKA        Go to that record
          00
704 F12E1 400         RTNC
705 F12E4 20          P=     =XchgT        Exchange buffers (talker)
706 F12E6 7310        GOSUB  Ddt
707 F12EA 7000        GOSUB  =ULYL         Address tape as listener
708 F12EE 20          P=     =CloseR       Close record (write buffer 0 out)
709 F12F0 7973        GOSUB  Ddl
710 F12F4 7D53        GOSUB  Tstat         Check tape status
711 F12F8 400         RTNC
712 F12FB 20   DdtXgT  P=     =XchgT        Exchange buffers back (talker)
713 F12FD 8C00 Ddt     GOLONG =DDT          Exit through DDT
          00
714                *-
715                *-
716 F1303 7453 PDIRBF  GOSUB  Mtyl          Address device as listener
717 F1307 400         RTNC                 Error
718 F130A 20          P=     =SetBP        Set byte pointer
719 F130C 7D53        GOSUB  Ddl
720 F1310 400         RTNC                 Error
721 F1313 AF9         C=B    W
722 F1316 7C53        GOSUB  Cslc4
723 F131A F2          CSL    A
724 F131C C6          C=C+C  A             C[B] is the byte pointer value
```

```
 725 F131E 7000          GOSUB  =Putd
 726 F1322 400           RTNC
 727 F1325 20            P=     =Write1        Write to buffer 1 of the device
 728 F1327 8C00          GOLONG =PUTDR"        Put out the directory entry.
           00
 729               *-
 730               *-
 731               *
 732               * Bug fix for pack (too many RSTK levels)
 733               *
 734 F132D 8E00 PACKfx  GOSUBL =GETDID
           00
 735 F1333 76BE          GOSUB  PDIR
 736 F1337 6210          GOTO   PACK00
 737               *-
 738               *-
 739 F133B 0             CON(1) =FIXSPC        1 nibble available here
 740               ****************************************************************
 741               ****************************************************************
 742               **
 743               ** Name:       PACK - Pack an HPIL mass storage device
 744               **
 745               ** Category:   STEXEC
 746               **
 747               ** Purpose:
 748               **      Pack an HPIL mass storage device
 749               **
 750               ** Entry:
 751               **      DO @ device spec
 752               **      P=0
 753               **
 754               ** Exit:
 755               **      Through NXTSTM...
 756               **
 757               ** Calls:      PDIR,GETDR",GT2BYT,GETZER,ASRC4,CSLC5,CSLC2,
 758               **             PT2BYT,PUTDR#,TSTAT,MOVEFL,NXTEN+,NXTEN-,GETDIR,
 759               **             ENDTAP,ASLC4,CSRC5,<NXTSTM>,<ERRORX>
 760               **
 761               ** Uses.......
 762               **  Inclusive: All CPU registers, STMTDO[3:0],STMTR1,FUNCxx,
 763               **             all RAM that EXPEXC is permitted to use
 764               **
 765               ** Stk lvls:   7 (PDIR)
 766               **
 767               ** Algorithm:
 768               **      GOSUB GETDID
 769               **      GOSUB PDIR
 770               **      Recall directory info from R0,R1
 771               **      Read and get directory entry
 772               **      IF end of directory THEN GOTO 9:
 773               **    2:IF file data area pointer <> PTRF THEN
 774               **         Copy file down, update directory
 775               **         Update file destination, read next entry
 776               **         GOTO 2:
 777               **      Get next directory entry
```

```
778                **      IF NOT end of directory THEN GOTO 2:
779                **    9:(end of directory)
780                **      Exit
781                **
782                ** History:
783                **
784                **    Date     Programmer          Modification
785                **  --------   ----------   --------------------------------
786                **  12/21/83      NZ        Changed call to PDIR to add a call
787                **                          to GETDID (RSTK level bug)
788                **  01/10/83      NZ        Rewrote routine and documentation
789                **
790                ****************************************************************
791                ****************************************************************
792 F133C 00000          REL(5) =PACKd
          0
793 F1341 00000          REL(5) =PACKp
          0
794 F1346 66EF =PACK     GOTO   PACKfx        Pack the directory first
795 F134A 493  PACK00    GOC    PACKeR        (error during directory pack)
796 F134D 118            C=R0                 Recall info from GDIRST
797 F1350 AF5            B=C    W
798                *
799                * Now B,R0[3:0] is PTRC...pointer to directory entry,
800                *     B,R0[7:4] is PTRF...pointer to data area
801                *
802 F1353 8E00 PACK10    GOSUBL =GETDR"       Get that directory entry
          00
803 F1359 4A2  PACK20    GOC    PACKeR        Error!
804                *
805                * Check for logical end of directory
806                *
807 F135C 173            D1=D1+ 4             Skip to type...
808 F135F 7F90           GOSUB  Gt2byt        Read 2 bytes...
809 F1363 E6             C=C+1  A             ...add 1 (if EOD, x0000)
810 F1365 F2             CSL    A             If EOD, 00000!
811 F1367 8AE            ?C#0   A             EOD?
812 F136A 60             GOYES  PACK30        No...continue
813 F136C 6970           GOTO   PACK90        Yes...done with the tape
814                *-
815                *-
816 F1370      PACK30
817                *
818                * Now D1 is positioned at the directory start address
819                *
820 F1370 7F70           GOSUB  GETZER        Read 2 bytes of zero, 2 of start
821 F1374 4F0            GOC    PACKeR        Error...out of range
822                *
823                * Now C[3:0] is the last 2 bytes of start
824                *
825 F1377 AF4            A=B    W
826 F137A 7000           GOSUB  =ASRC4        Now PTRF in A[3:0], A[4]=0
827 F137E D5             B=C    A             Copy start to B[A]
828 F1380 7F60           GOSUB  GETZER        Read 2 bytes of zero, 2 of size
829 F1384 4F4  PACKeR    GOC    PACKer        Error...out of range
```

```
830             *
831             * Now C[A] is size of file in sectors
832             *
833 F1387 8A0           ?A=B   A             Is the file already in place?
834 F138A E4            GOYES  PACK40        Yes...continue
835             *
836             * Need to move the file data
837             *
838 F138C 73E2          GOSUB  Cslc5
839 F1390 D6            C=A    A             C[A] is dest, C[9:5] is length
840 F1392 10B           R3=C
841             *
842             * A[A],R3[A] is dest, B[A] is source, R3[9:5] is length,
843             * R2[A] is the source address, R1[A] is dest address
844             *
845 F1395 1CB           D1=D1- 12            Back up to middle of start addr
846 F1398 D6            C=A    A
847 F139A 7000          GOSUB  =CSLC2
848 F139E 8E00          GOSUBL =PT2BYT       Write 2 bytes @ D1
          00
849             *
850             * Now update the directory entry in the directory
851             *
852 F13A4 118           C=R0
853 F13A7 816           CSRC
854 F13AA AD2           C=0    M             Now C[S] is entry, C[X] is addr
855 F13AD 8E00          GOSUBL =PUTDR#       Write the entry to the device
          00
856 F13B3 402           GOC    PACKer        Error
857 F13B6 7B92          GOSUB  Tstat         Check status
858 F13BA 491           GOC    PACKer        Error
859 F13BD 119           C=R1
860 F13C0 10A           R2=C                 Copy address to R1,R2 for MOVEFL
861             *
862             * A,C,D and R4 are available to MOVEFL...
863             *
864 F13C3 8E00          GOSUBL =MOVEFL       Move file
          00
865 F13C9 4A0           GOC    PACKer        Error
866             *
867             * Nxten+ does not return if an error occurs
868             *
869 F13CC 7830          GOSUB  Nxten+        Go to next entry...
870 F13D0 628F          GOTO   PACK10        ...and continue loop if return
871             *-
872             *-
873 F13D4 6032 PACKer   GOTO   Errorx
874             *-
875             *-
876 F13D8       PACK40
877             *
878             * This entry is OK where it is now
879             *
880             * A[A] is PTRF, C[A] is file length
881             *
```

```
882 F13D8 7A30          GOSUB  Nxten-        Increment to next entry
883 F13DC 8E00          GOSUBL =GETDIR       Read the next directory entry
          00
884 F13E2 667F          GOTO   PACK20        Check error @ PACK20
885             *-
886             *-
887             *
888             * If here, reached end of directory
889             *
890 F13E6 8E00 PACK90   GOSUBL =ENDTAP       Clean the device up (rewind, etc)
          00
891 F13EC 47E           GOC    PACKer        Error
892 F13EF 6093          GOTO   nXTSTM        No error...exit
893             *-
894             *-
895 F13F3 D2   =GETZER  C=0    A
896 F13F5 7900          GOSUB  Gt2byt
897 F13F9 8AA           ?C=0   A
898 F13FC 60            GOYES  Gt2byt
899 F13FE 20            P=     =eRANGE
900 F1400 02            RTNSC
901             *-
902             *-
903 F1402 8C00 Gt2byt   GOLONG =GT2BYT
          00
904             *-
905             *-
906 F1408 110  Nxten+   A=R0
907 F140B 7000          GOSUB  =ASRC4        Get file start address
908 F140F 11B           C=R3
909 F1412 7462          GOSUB  Csrc5         Get length of file into C[A]
910 F1416 23   Nxten-   P=     3
911 F1418 A1A           A=A+C  WP            Add length to start of file
912 F141B 20            P=     =eRANGE
913 F141D 46B           GOC    PACKer        Error if carry
914 F1420 7000          GOSUB  =ASLC4        Return to proper location
915 F1424 D6            C=A    A
916 F1426 8E00          GOSUBL =NXTENT
          00
917 F142C DA            A=C    A
918 F142E 100           R0=A
919 F1431 AF8           B=A    W             Copy to B[W] too
920 F1434 500           RTNNC                If no carry, same entry
921 F1437 119           C=R1
922 F143A 7C32          GOSUB  Csrc5
923 F143E CE            C=C-1  A             Decrement counter
924 F1440 7F22          GOSUB  Cslc5
925 F1444 109           R1=C
926 F1447 4E9           GOC    PACK90        If carry set, EOD (RSTK=garbage)
927 F144A 03            RTNCC                Not at EOD yet...continue
928             ********************************************************************
929             ********************************************************************
930             **
931             ** Name:       INITXQ - Execute the INITIALIZE statement
932             **
```

```
933                ** Category:   STEXEC
934                **
935                ** Purpose:
936                **      Initialize the specified mass storage device's medium
937                **
938                ** Entry:
939                **      D0 points to the device specifier
940                **
941                ** Exit:
942                **      If error, exits through ERRORX;
943                **      If no error, exits through ENDST
944                **
945                ** Calls:      GETPIL,SAVE2C,TRESD0,SAVED1,SAVE1A,GETHEX,RESTD1,
946                **             REST2C,ASRC4,REST1A,START,CHKMAS,FORMAT,<ENDST>,
947                **             <ERRORX>
948                **
949                ** Uses.......
950                **  Inclusive: All CPU registers, STMTD1,STMTRx,FUNCxx,ST[11:0],
951                **             all RAM EXPEXC is permitted to use,SCRTCH[63:0]
952                **
953                ** Stk lvls:   7 (GETPIL)
954                **
955                ** History:
956                **
957                **    Date     Programmer          Modification
958                **  --------   ----------   --------------------------------
959                **  11/29/83      NZ        Updated documentation
960                **  12/15/82      NZ        Updated documentation
961                **
962                ********************************************************************
963                ********************************************************************
964 F144C 0000          REL(5) =INITd         INITIALIZE decompile
           0
965 F1451 0000          REL(5) =INITp         INITIALIZE parse
           0
966 F1456          =INITXQ
967                *
968                * Get the file specifier (volume label, device spec)
969                *
970 F1456 8E00          GOSUBL =GETPIL
           00
971 F145C 4F4           GOC    INITXF         Error
972                *
973                * Now B[W] is the device type or word, D[X] is device address,
974                * R0 is the volume label, C[6:0] is the recall word from SETUP
975                *
976 F145F 7E12          GOSUB  Save2c         Save recall word in STMTR1
977 F1463 7832          GOSUB  TresD0         Get PC from FUNCD0 (from GETPIL)
978 F1467 20            P=     0
979 F1469 3100          LC(2)  =tCOMMA
980 F146D 14A           A=DAT0 B
981 F1470 962           ?A=C   B              # entries specified?
982 F1473 51            GOYES  INITX0         Yes...skip the comma first
983                *
984                * Number of entries not specified...use default length
```

```
 985                *
 986 F1475 110          A=RO                    Length field is RO[15:12]
 987 F1478 2C           P=      12              Clear nibbles 12-15
 988 F147A A80  INITLP  A=0     P
 989 F147D 0C           P=P+1
 990 F147F 5AF          GONC    INITLP
 991 F1482 100          RO=A                    Put new vol label, length into RO
 992 F1485 426          GOC     INITX1          Go always
 993                *-
 994                *-
 995                *
 996                * Found a comma (number of entries specified)
 997                *
 998 F1488 161  INITXO  DO=DO+ 2                Skip the comma
 999 F148B DB           C=D     A               Save D[A] in STMTD1
1000 F148D 135          D1=C
1001 F1490 8E00         GOSUBL =SAVED1          Save device address in STMTD1
          00
1002 F1496 118          C=RO
1003 F1499 74E1         GOSUB  Save2c           Save volume label in STMTR1
1004 F149D AF4          A=B     W
1005 F14A0 8E00         GOSUBL =SAVE1A          Save device word in STMTRO
          00
1006 F14A6 8E00         GOSUBL =GETHEX          Get # of entries (4 nibs max)
          00
1007 F14AC 4C5  INITXF  GOC     INITXE          Error in expression evaluation
1008               *
1009 F14AF 20           P=      =eRANGE         Check if valid range
1010 F14B1 8A8          ?A=0    A               Is the value zero?
1011 F14B4 8F           GOYES   INITXF          Yes...error
1012 F14B6 8E00         GOSUBL =RESTD1          Restore device address to D[A]
          00
1013 F14BC 137          CD1EX
1014 F14BF D7           D=C     A
1015 F14C1 8E00         GOSUBL =REST2C          Restore volume label to RO
          00
1016 F14C7 108          RO=C
1017 F14CA 7000         GOSUB  =ASRC4           Rotate value into A[15:12]
1018 F14CE AF8          B=A     W               Save value in B[15:12] for now
1019 F14D1 8E00         GOSUBL =REST1A          Restore device word to A[W]
          00
1020               *
1021               * Now A[W] is device word, B[15:12] is # of entries, RO is vol
1022               * label, D[A] is device address
1023               *
1024               * Combine volume label and # of entries in RO
1025               *
1026 F14D7 120          AROEX
1027 F14DA 2B           P=      11
1028 F14DC A9C          ABEX    WP              Volume label in B[11:0]
1029 F14DF AFC          ABEX    W               Volume label, # entries in A
1030 F14E2 120          AROEX                   Volume label->RO, device word->A
1031 F14E5 AF8          B=A     W               Device word back in B[W]
1032               *
1033 F14E8 8E00 INITX1  GOSUBL =START           Set up the loop, find the device
```

```
            00
 1034 F14EE 4A1          GOC    INITXE        Error
 1035 F14F1 8E00         GOSUBL =CHKMAS       Check if mass storage (must be!)
            00
 1036 F14F7 411          GOC    INITXE        Error
 1037               *
 1038               * It is mass storage...OK to continue
 1039               *
 1040 F14FA 8E00         GOSUBL =FORMAT       Format the medium, initialize fields
            00
 1041 F1500 480          GOC    INITXE        Error
 1042 F1503 6C31         GOTO   Endst         No error...clean up, exit
 1043               *-
 1044               *-
 1045               *
 1046               * Following line is never referenced!(?)
 1047               *
 1048 F1507 20   INITX2  P=     =eDTYPE       Device type error
 1049 F1509 6BF0 INITXE  GOTO   Errorx
 1050               ****************************************************************
 1051               ****************************************************************
 1052               **
 1053               ** Name:       LOCAL - Execute the LOCAL [LOCKOUT] statement
 1054               **
 1055               ** Category:   STEXEC
 1056               **
 1057               ** Purpose:
 1058               **      LOCAL statement sends a NRE to entire loop, or a GTL
 1059               **      frame to devices specified.  LOCAL LOCKOUT sends
 1060               **      a LLO frame to loop specified.
 1061               **
 1062               ** Entry:
 1063               **      DO points to the token following LOCAL
 1064               **
 1065               ** Exit:
 1066               **      Through CLEARc
 1067               **
 1068               ** Calls:      <CLEARc>
 1069               **
 1070               ** Uses.......
 1071               **  Inclusive: Same as CLEARc
 1072               **
 1073               ** Stk lvls:   Same as CLEARc
 1074               **
 1075               ** History:
 1076               **
 1077               **    Date     Programmer             Modification
 1078               **  --------   ----------    ---------------------------------
 1079               **  01/25/83      JH         Added Routine
 1080               **
 1081               ****************************************************************
 1082               ****************************************************************
 1083 F150D 0000         REL(5) =LOCALd
            0
 1084 F1512 0000         REL(5) =LOCALp
```

```
           0
1085 F1517      =LOCAL
1086            *
1087            *  Is the next token LOCKOUT?
1088            *
1089 F1517 AFA          A=C    W            (Copy high nibs for compare)
1090 F151A 15A5         A=DAT0 6            Read next token
1091            *****
1092            *
1093            *       LC(6)  (=tLOCKO)~(=LEXPIL)~(=tXWORD)
1094 F151E 35           NIBHEX 35           LC(6)
1095 F1520 00           CON(2) =tXWORD      ...
1096 F1522 00           CON(2) =LEXPIL      ..
1097 F1524 00           CON(2) =tLOCKO      .
1098            *
1099            *****
1100 F1526 976          ?A#C   W            LOCAL LOCKOUT statement?
1101 F1529 D1           GOYES  LCL10        No...execute LOCAL statement
1102 F152B 7161         GOSUB  D1=SDO       Yes...set up LLO frame
1103 F152F 3411         LC(5)  #11~#11      Set C[3:0] to value of LLO frame
           110
1104 F1536 145          DAT1=C A            Save frame in STMTDO
1105 F1539 165          DO=DO+ 6            Skip the LOCKOUT token
1106 F153C 8E00         GOSUBL =CKLOP#      Get the loop # to C[S]
           00
1107 F1542 6F50         GOTO   CLEAR1       Continue with loop
1108            *-
1109            *-
1110 F1546 3410 LCL10   LC(5)  #93~#01      Set C[3:0] to NRE and GTL frames
           390
1111 F154D 6E30         GOTO   CLEARc       Execution same as CLEAR
1112            ********************************************************************
1113            ********************************************************************
1114            **
1115            ** Name:       TRIGGER - Execute the TRIGGER statement
1116            **
1117            ** Category:   STEXEC
1118            **
1119            ** Purpose:
1120            **      Sends a GET to entire loop, or devices specified
1121            **      are addressed to listen and then GET is sent.
1122            **
1123            ** Entry:
1124            **      DO points to the token following TRIGGER
1125            **
1126            ** Exit:
1127            **      Through CLEARc
1128            **
1129            ** Calls:      Same as CLEARc
1130            **
1131            ** Uses.......
1132            **  Inclusive: Same as CLEARc
1133            **
1134            ** Stk lvls:   Same as CLEARc
1135            **
```

```
1136            ** History:
1137            **
1138            **    Date     Programmer          Modification
1139            **  --------   ----------   --------------------------------
1140            **  01/25/83      JH        Added routine
1141            **
1142            ********************************************************************
1143            ********************************************************************
1144 F1551 0000        REL(5) =TRIGd
            0
1145 F1556 0000        REL(5) =TRIGp
            0
1146 F155B      =TRIGER                     Set C[3:0] to values of GET and
1147 F155B 3480        LC(5)  #08~#08       GET frame
            800
1148 F1562 6920        GOTO   CLEARc        Execute same as CLEAR
1149            ********************************************************************
1150            ********************************************************************
1151            **
1152            ** Name:       REMOTE - Execute the REMOTE statement
1153            **
1154            ** Category:   STEXEC
1155            **
1156            ** Purpose:
1157            **      Sends an UNL, RFC, REN, RFC, then addresses the device
1158            **      specified, if any, as listener
1159            **
1160            ** Entry:
1161            **      DO points to the token following REMOTE
1162            **
1163            ** Exit:
1164            **      Through CLEARc
1165            **
1166            ** Calls:      Same as CLEARc
1167            **
1168            ** Uses.......
1169            **  Inclusive: Same as CLEARc
1170            **
1171            ** Stk lvls:   Same as CLEARc
1172            **
1173            ** History:
1174            **
1175            **    Date     Programmer          Modification
1176            **  --------   ----------   --------------------------------
1177            **  03/19/83      NZ        Rewrote routine and documentation
1178            **  01/26/83      JH        Added routine
1179            **
1180            ********************************************************************
1181            ********************************************************************
1182 F1566 0000        REL(5) =REMOTd
            0
1183 F156B 0000        REL(5) =REMOTp
            0
1184 F1570 3429 =REMOTE LC(5)  #F9292       Set the REMOTE flag, REN~REN
            29F
```

```
1185 F1577 6410          GOTO   CLEARc
1186              ****************************************************************
1187              ****************************************************************
1188              **
1189              ** Name:       CLEAR - Execute the CLEAR statement
1190              ** Name:       CLEARc - Execute a loop statement
1191              **
1192              ** Category:   STEXEC
1193              **
1194              ** Purpose:
1195              **      Execute the CLEAR statement (also TRIGGER, LOCAL,
1196              **      REMOTE)
1197              **
1198              ** Entry:
1199              **      D0 points to the device specifier
1200              **      CLEARc: C[3:0] is the 2 frames, C[4] is REMOTE flag-
1201              **       "F" means REMOTE, "0" means other
1202              **
1203              ** Exit:
1204              **      Through ENDST if no error, through ERRORX if error
1205              **
1206              ** Calls:      D1=SR0,FNDCH-,GETDID,CKmode,UNLPUT,PUTC,PRTISc,
1207              **             SAVEIT,D1=SD0,GETMBX,<ENDST>,<ERRORX>
1208              **
1209              ** Uses.......
1210              **  Inclusive: All CPU registers, STMTDx,STMTR1,FUNCxx,ST[11:0],
1211              **             all RAM EXPEXC is permitted to use
1212              **
1213              ** Stk lvls:   7 (GETDID)
1214              **
1215              ** History:
1216              **
1217              **    Date     Programmer          Modification
1218              **  --------   ----------   --------------------------------
1219              **  04/05/83      NZ        Moved controller check to include
1220              **                          case of device spec given
1221              **  03/19/83      NZ        Rewrote routine and documentation
1222              **
1223              ****************************************************************
1224              ****************************************************************
1225 F157B 0000        REL(5) =CLEARd
          0
1226 F1580 0000        REL(5) =CLEARp
          0
1227 F1585 3440 =CLEAR LC(5)  #14~#04       DCL ~ SDC frames (high nib=0)
          410
1228 F158C 7001 CLEARc GOSUB  D1=SD0        Save C[3:0] in STMTD0 (frames)
1229 F1590 145         DAT1=C A
1230 F1593 14A         A=DAT0 B             Check if there is a device spec
1231 F1596 3100        LC(2)  =tCOMMA       (tCOMMA means no device spec)
1232 F159A 966         ?A#C   B
1233 F159D 22          GOYES  CLEAR.        No device spec...use LOOP
1234 F159F AC2         C=0    S             Use loop 0 if none given
1235 F15A2       CLEAR1
1236 F15A2 AC7         D=C    S             Save mailbox # for later...
```

```
1237 F15A5 8E00          GOSUBL =FNDCH-       Find that mailbox
           00
1238 F15AB 495           GOC    Errorx        Error if carry
1239 F15AE 813           DSLC                 Mailbox # to D[0]
1240 F15B1 F3            DSL    A
1241 F15B3 F3            DSL    A             Mailbox # to D[2]
1242 F15B5 C7            D=D+D  A
1243 F15B7 C7            D=D+D  A             (Mailbox #)*16 to D[2]
1244 F15B9 DB            C=D    A             Device = :LOOP:#
1245 F15BB 6C00          GOTO   CLEAR+        (Carry not sure)
1246               *-
1247               *-
1248 F15BF       CLEAR.
1249               *
1250               * Device spec follows...get it
1251               *
1252 F15BF 8E00          GOSUBL =GETDID       Get device spec, don't check MS
           00
1253 F15C5 4F3           GOC    Errorx        Error
1254 F15C8 8E00 CLEAR+   GOSUBL =CKmode       Controller check (exits to error)
           00
1255 F15CE 75B0          GOSUB  D1=SR0        Save device spec in STMTR0
1256 F15D2 15D6          DAT1=C 7             (Write it out)
1257 F15D6 1D00          D1=(2) (=STMTD0)+2   Check if REMOTE
1258 F15DA 15F2          C=DAT1 3             Read frame, flag (C[XS] is flag)
1259 F15DE 92A           ?C=0   XS            Is it REMOTE?
1260 F15E1 21            GOYES  CLEAR1        No...continue
1261               *
1262               * This is the REMOTE statement
1263               *
1264 F15E3 8E00          GOSUBL =UNLPUT       Send the UNL command
           00
1265 F15E9 4B1           GOC    Errorx        Error if carry
1266 F15EC 7650          GOSUB  CLEARs        Send the frame @ D1
1267 F15F0 441           GOC    Errorx        Error if carry
1268 F15F3 7090 CLEAR1   GOSUB  D1=SR0        Set D1 @ STMTR0 (device spec)
1269 F15F7 821           XM=0                 Clear XM to detect error - PRTISc
1270 F15FA 7C99          GOSUB  PRTISc        Address the device as listener
1271 F15FE 831           ?XM=0                OK?
1272 F1601 A0            GOYES  CLEAR2        Yes...continue
1273               *
1274               * Error detected by PRTISc...must be loop error!?
1275               *
1276 F1603 20            P=     =eABORT       Set eABORT to check it in ERRORX
1277 F1605 8C00 Errorx   GOLONG =ERRORX
           00
1278               *-
1279               *-
1280 F160B       CLEAR2
1281               *
1282               * Now release any I/O buffers created by GETDID
1283               *
1284 F160B 7870          GOSUB  D1=SR0
1285 F160F 147           C=DAT1 A             Read the device spec for below
1286 F1612 06            RSTK=C               Save device spec on stack
```

```
1287 F1614 AF2          C=0    W
1288 F1617 7640         GOSUB  Saveit        SAVEIT will release any buffers
1289 F161B 07           C=RSTK               Pop device spec...
1290 F161D DA           A=C    A             ...and put in A[A]
1291 F161F 7D60         GOSUB  D1=SDO        Set D1 @ STMTDO
1292 F1623 147          C=DAT1 A
1293 F1626 C6           C=C+C  A             Check if REMOTE (Carry if so)
1294 F1628 471          GOC    CLEAR4        REMOTE...exit
1295 F162B 8E00         GOSUBL =GETMBX       Get the mailbox address back
          00
1296               *
1297               * A[A] contains the device spec, D1 @ STMTDO
1298               *
1299 F1631 96C          ?A#0   B             Was the device LOOP?
1300 F1634 50           GOYES  CLEAR3        No...REAL device
1301 F1636 171          D1=D1+ 2             Yes...use the LOOP spec
1302 F1639 7900 CLEAR3  GOSUB  CLEARs        Send the command
1303 F163D 47C          GOC    Errorx        Error if carry
1304 F1640      CLEAR4
1305 F1640 8C00 Endst   GOLONG =ENDST        Done
          00
1306               *-
1307               *-
1308 F1646 3300 CLEARs  LC(4)  =mCMDf        Send the command frame...
          00
1309 F164C 14F          C=DAT1 B             ...from @ D1
1310 F164F 8C00 Putc    GOLONG =PUTC
          00
1311               *-
1312               *-
1313 F1655 8C00 Tstat   GOLONG =TSTAT
          00
1314               *-
1315               *-
1316 F165B 8C00 Mtyl    GOLONG =MTYL
          00
1317               *-
1318               *-
1319 F1661 8C00 Saveit  GOLONG =SAVEIT
          00
1320               *-
1321               *-
1322 F1667 8C00 Nxtchr  GOLONG =NXTCHR
          00
1323               *-
1324               *-
1325 F166D 8C00 Ddl     GOLONG =DDL
          00
1326               *-
1327               *-
1328 F1673 812  Cslc5   CSLC                 Fall into CSLC4
1329 F1676 6000 Cslc4   GOTO   =CSLC4
1330               *-
1331               *-
1332 F167A 816  Csrc5   CSRC                 Fall into CSRC4!
```

```
1333 F167D 6000 Csrc4   GOTO   =CSRC4
1334            *-
1335            *-
1336 F1681 8C00 Save2c  GOLONG =SAVE2C
           00
1337            *-
1338            *-
1339 F1687 1F00 =D1=SR0 D1=(5) =STMTR0
           000
1340 F168E 01           RTN
1341            *-
1342            *-
1343 F1690 1F00 =D1=SD0 D1=(5) =STMTD0
           000
1344 F1697 01           RTN
1345            *-
1346            *-
1347 F1699 8C00 TsavdO  GOLONG =TSAVD0
           00
1348            *-
1349            *-
1350 F169F 8C00 TresdO  GOLONG =TRESD0
           00
1351            ****************************************************************
1352            ****************************************************************
1353            **
1354            ** Name:       STANBY - Execute the STANDBY statement
1355            **
1356            ** Category:   STEXEC
1357            **
1358            ** Purpose:
1359            **      Execute the standby statement
1360            **
1361            ** Entry:
1362            **      D0 points to the first parameter
1363            **
1364            ** Exit:
1365            **      Through NXTSTM if no error, ERRORX if error
1366            **
1367            ** Calls:      GLOOP#,SAVE2C,STANsb,REST2C,IDIV,FNDCHK,PUTC,
1368            **             PUTE,<NXTSTM>
1369            **
1370            **      STANsb:EXPEXC,POP1N,FLTDH
1371            **
1372            ** Uses.......
1373            **  Inclusive: All CPU registers,STMTR1,FUNCxx,ST[11:0],all
1374            **             RAM that EXPEXC is permitted to use
1375            **
1376            ** Stk lvls:   7 (GLOOP#)
1377            **
1378            ** History:
1379            **
1380            **    Date     Programmer          Modification
1381            **  --------   ----------   ---------------------------------
1382            **  05/18/83      NZ        Changed # of IDY timeouts (+1)...
```

```
1383                **                              due to user misunderstanding
1384                **  03/21/83      NZ            Changed CHECKC to inline code
1385                **  02/25/83      NZ            Wrote, added documentation
1386                **
1387                ****************************************************************
1388                ****************************************************************
1389 F16A5 0000        REL(5) =STANDd       Standby decompile
          0
1390 F16AA 0000        REL(5) =STANDp       Standby parse
          0
1391 F16AF 8E00 =STANBY GOSUBL =GLOOP#       Get loop # to C[S]
          00
1392 F16B5 AC7          D=C    S             Save in D[S]
1393 F16B8 14A          A=DAT0 B             Read next token
1394 F16BB 3100         LC(2)  =tOFF         Check if "STANDBY OFF"
1395 F16BF 962          ?A=C   B             Is it "OFF"?
1396 F16C2 11           GOYES  STAN10        Yes...set up the values
1397 F16C4 3100         LC(2)  =tON          Check if "ON"
1398 F16C8 966          ?A#C   B             Is it "ON"?
1399 F16CB 02           GOYES  STAN20        No...must be numeric values
1400                *
1401                * This is "STANDBY ON"
1402                *
1403 F16CD D3           D=0    A             Set frame timeout=0
1404 F16CF 6480         GOTO   STAN40
1405                *-
1406                *-
1407 F16D3      STAN10
1408                *
1409                * This is "STANDBY OFF"
1410                *
1411 F16D3 3400         LC(5)  =Timout       Frame timeout value
          000
1412 F16DA D7           D=C    A             Put in D[A]
1413 F16DC 3100         LC(2)  =#Timeo       # of IDY timeouts
1414 F16E0 D5           B=C    A             Put in B[B]
1415 F16E2 417          GOC    STAN40        Go always
1416                *-
1417                *-
1418 F16E5 20   STANra  P=     =eRANGE       Arg out of range
1419 F16E7 6D1F STANer  GOTO   Errorx        Error
1420                *-
1421                *-
1422 F16EB      STAN20
1423                *
1424                * This is STANDBY <expr> [,<expr>]
1425                *
1426                * Evaluate the frame timeout after saving loop #
1427                *
1428 F16EB ACB          C=D    S             Recall loop # to C[S]
1429 F16EE 7F8F         GOSUB  Save2c        Save in STMTR1[S]
1430 F16F2 7790         GOSUB  STANsb        Manipulate frame timeout
1431 F16F6 40F          GOC    STANer        Error if carry
1432 F16F9 8E00         GOSUBL =REST2C       Restore loop # to C[S]
          00
```

```
1433                *
1434                * A[A] is now the timeout value
1435                *
1436 F16FF D6             C=A    A
1437 F1701 D7            D=C    A              Put timeout value in D[A]
1438 F1703 D1            B=0    A              Clear B[B] (# of IDY timeouts)
1439 F1705 E5            B=B+1  A              (# of IDY timeouts: 0=infinity)
1440 F1707 767F          GOSUB  Save2c         Timeout in STMTR1,loop # in [S]
1441 F170B 14A           A=DAT0 B
1442 F170E 3100          LC(2)  =tCOMMA
1443 F1712 966           ?A#C   B              Is there a comma?
1444 F1715 F3            GOYES  STAN40         No...use default (same as first)
1445 F1717 161           D0=D0+ 2              Comma...skip it
1446                *
1447                * Read the IDY timeout value
1448                *
1449 F171A 17F           D1=D1+ 16             Remove the first entry from stack
1450                *
1451                * Now evaluate IDY timeout
1452                *
1453 F171D 7C60          GOSUB  STANsb         Evaluate expr, massage it
1454 F1721 45C  STANeR   GOC    STANer         Error
1455                *
1456                * A[A] is now the IDY timeout
1457                *
1458 F1724 D6            C=A    A
1459 F1726 D7            D=C    A              Set D[A] to IDY timeout
1460 F1728 8E00          GOSUBL =REST2C        Restore frame timeout to C[A]
          00
1461 F172E AC7           D=C    S              Restore loop #
1462 F1731 AF0           A=0    W
1463 F1734 DA            A=C    A              A[W] is now frame timeout
1464 F1736 AF2           C=0    W
1465 F1739 DB            C=D    A              C[W] is now IDY timeout
1466 F173B 8F00          GOSBVL =IDIV
          000
1467                *
1468                * Now A[W] is quotient, B,C[W] are remainder
1469                *
1470 F1742 97A           ?C=0   W
1471 F1745 50            GOYES  STAN30         Exact multiple...OK
1472 F1747 B74           A=A+1  W              Remainder...round up
1473 F174A D8   STAN30   B=A    A              Copy count to B[B]
1474 F174C AE0           A=0    B              Check if too many IDY timeouts
1475 F174F 97C           ?A#0   W              In range?
1476 F1752 39            GOYES  STANra         No...range error
1477 F1754 20   STAN40   P=     0
1478                *
1479                * Now D[A] is timeout value, B[B] is # IDY timeouts, D[S] is
1480                * loop #
1481                *
1482 F1756 ACB           C=D    S
1483 F1759 7386          GOSUB  Fndchk         Find the mailbox (C[S]=loop #)
1484 F175D 43C           GOC    STANeR         Error...not found or man mode
1485 F1760 3300          LC(4)  =mSETIC        Set number of IDY timeouts...
```

```
            00
1486 F1766 AE9          C=B     B               ...to B[B]
1487 F1769 72EE         GOSUB   Putc
1488 F176D 43B          GOC     STANeR          Error...abort
1489 F1770 25           P=      5
1490 F1772 300          LC(1)   =mSTO@5         Set frame timeout...
1491 F1775 DB           C=D     A               ...to D[A]
1492 F1777 8E00         GOSUBL  =PUTE
            00
1493 F177D 43A          GOC     STANeR          Error...abort
1494 F1780 8D00 =nXTSTM GOVLNG  =NXTSTM         Done
            000
1495                    *-
1496                    *-
1497 F1787 8C00 Pop1n   GOLONG  =POP1N
            00
1498                    *-
1499                    *-
1500 F178D 8E00 STANsb  GOSUBL  =eXPEXC         Evaluate the expression
            00
1501 F1793 70FF         GOSUB   Pop1n           Pop it off the stack
1502 F1797 400          RTNC                    Error
1503                    *
1504                    * Multiply by 1000 (convert to millisecs)
1505                    *
1506 F179A 3230         LC(3)   3               10^3 is 1000
            0
1507 F179F 05           SETDEC
1508 F17A1 A3A          A=A+C   X               Can't be shortened to A field
1509 F17A4 D6           C=A     A               Check if still negative...
1510 F17A6 A36          C=C+C   X
1511 F17A9 04           SETHEX
1512 F17AB 401          GOC     STANsr          Range error if carry
1513 F17AE 7B77         GOSUB   fLTDH           Convert to HEX
1514 F17B2 590          GONC    STANsr          Out of range or data type
1515 F17B5 8A8          ?A=0    A               Zero is NOT valid for timeout
1516 F17B8 40           GOYES   STANsr
1517 F17BA 03           RTNCC                   Good data...return
1518                    *-
1519                    *-
1520 F17BC 20   STANsr  P=      =eRANGE         Out of range
1521 F17BE 02           RTNSC
1522                    ********************************************************************
1523                    ********************************************************************
1524                    **
1525                    ** Name:        LISTIO - Execute the LIST IO statement
1526                    **
1527                    ** Category:    STEXEC
1528                    **
1529                    ** Purpose:
1530                    **      LIST IO user statement: list the devices in the ASSIGN
1531                    **      IO table (if none, error)
1532                    **
1533                    ** Entry:
1534                    **      P=0
```

```
1535                    **
1536                    ** Exit:
1537                    **      Through NXTSTM if no error, through ERRORX if error
1538                    **
1539                    ** Calls:      I/OFND,HTOD,D1=SDO,BLANKC,WRTASC,BF2DSP,ASLC2,
1540                    **             ASRC2,<NXTSTM>,<ERRORX>
1541                    **
1542                    ** Uses.......
1543                    **  Inclusive: A-D,R3,ST[11:0],STMTxx,FUNCxx
1544                    **
1545                    ** Stk lvls:   5 (BF2DSP)
1546                    **
1547                    ** History:
1548                    **
1549                    **    Date      Programmer            Modification
1550                    **  --------    ----------    --------------------------------
1551                    **  01/16/84       NZ         Fixed device # count to count in
1552                    **                            DECIMAL, not HEX!
1553                    **  12/15/82       NZ         Updated documentation
1554                    **
1555                    ****************************************************************
1556                    ****************************************************************
1557 F17C0 300  LISTnb  LC(1)  =eNOASN       "ASSIGN IO Needed"
1558 F17C3 20           P=     =ePARSE       (parse message)
1559 F17C5 6F3E         GOTO   Errorx
1560                    *-
1561                    *-
1562 F17C9 0000         REL(5) =OFFIOd       IO decompile
          0
1563 F17CE 0000         REL(5) =IOp          IO parse
          0
1564 F17D3       =LISTIO
1565 F17D3 3200         LC(3)  =bPILAI       Assign IO buffer
          0
1566 F17D8 8E00         GOSUBL =i/OFND
          00
1567 F17DE 51E          GONC   LISTnb        No buffer...error
1568 F17E1 AF2          C=0    W             Clear nibs 14 & 15
1569 F17E4 137          CD1EX
1570 F17E7 134          DO=C
1571 F17EA 135          D1=C
1572 F17ED 10B          R3=C                 Save buffer pointer in R3
1573                    *
1574                    * Now DO,D1 point to the ASSIGN IO buffer
1575                    *
1576                    * First figure out how many devices ARE assigned
1577                    *
1578 F17F0 D1           B=0    A             B[A] is the device count
1579 F17F2 E5   LIST10  B=B+1  A             increment count
1580 F17F4 147          C=DAT1 A             Read this entry
1581 F17F7 173          D1=D1+ 4
1582 F17FA 96E          ?C#0   B             Is this entry null?
1583 F17FD 5F           GOYES  LIST10        No...continue
1584                    *
1585                    * Now B[X] is the device count
```

```
1586                *
1587 F17FF CD              B=B-1  A            Back off last count (the null)
1588 F1801 D9              C=B    A            Copy count to C[X]
1589 F1803 8E00            GOSUBL =HTOD        Convert to decimal
           00
1590 F1809 04              SETHEX              Now B[B] is the decimal value
1591 F180B 718E            GOSUB  D1=SDO       Use STMTDx,FUNCxx to write it
1592 F180F 8E00            GOSUBL =BLANKC      Set C[W]="        "
           00
1593 F1815 15D5            DAT1=C 6            Write out 3 blanks
1594 F1819 21              P=     1            Two digits (B[B])
1595 F181B AC3             D=0    S            CLEAR the sign for WRTASC
1596 F181E 8E00            GOSUBL =WRTASC      WRiTe ASCii
           00
1597 F1824 171             D1=D1+ 2            Leave a blank after the number
1598 F1827 20              P=     0
1599 F1829 3F44            LCASC  \s(eciveD\   "Device(s"
           5667
           9636
           5682
           37
1600 F183B 1557            DAT1=C W
1601 F183F 17F             D1=D1+ 16
1602 F1842 3F92            LCASC  \ngissa )\   ") assign"
           0216
           3737
           9676
           E6
1603 F1854 1557            DAT1=C W
1604 F1858 17F             D1=D1+ 16
1605                *      LC(10) #FF0A0D*(#10000)+\de\ "ed"&Cr&Lf&chr$(255)
1606 F185B 39              NIBHEX 39
1607 F185D 5646            NIBASC \ed\
1608 F1861 D0A0            NIBHEX D0A0FF
           FF
1609                *
1610 F1867 15D9            DAT1=C 10
1611 F186B 1D00            D1=(2) =STMTD0
1612 F186F 8F00            GOSBVL =BF2DSP      Send to display, ignore width
           000
1613                *
1614                * Now have sent the header
1615                *
1616                * Send out lines until reach a zero byte
1617                *
1618 F1876 3F44 LIST20     LCASC  \# eciveD\   "Device #"
           5667
           9636
           5602
           32
1619 F1888 1F00            D1=(5) =FUNCR0
           000
1620 F188F 1557            DAT1=C W
1621 F1893 17F             D1=D1+ 16
1622 F1896 3302            LCASC  \  \
```

```
            02
1623 F189C 15D3         DAT1=C 4            Write blanks out to initialize
1624 F18A0 113          A=R3                Get buffer address, counter
1625 F18A3 8E00         GOSUBL =ASLC5
            00
1626 F18A9 05           SETDEC              Increment in DECIMAL mode
1627 F18AB E4           A=A+1  A            Increment A[B]
1628 F18AD 04           SETHEX              Return to HEX mode
1629 F18AF D8           B=A    A            Copy to B[B]
1630 F18B1 8E00         GOSUBL =ASRC5
            00
1631 F18B7 130          D0=A                Set D0 @ buffer
1632 F18BA 163          D0=D0+ 4
1633 F18BD 132          ADOEX               D0 @ entry, A[A] @ next entry
1634 F18C0 103          R3=A                Store new count in R3
1635 F18C3 21           P=     1            Write B[B]
1636 F18C5 AC3          D=0    S            Sign is positive
1637 F18C8 8E00         GOSUBL =WRTASC      Write ASCII @ D1
            00
1638 F18CE 20           P=     0
1639 F18D0 35D3         LCASC  \:'=\        ":'="
            72A3
1640 F18D8 15D5         DAT1=C 6
1641 F18DC 175          D1=D1+ 6
1642            *
1643            * Now read the 2 letters, put them in RAM, display them
1644            *
1645 F18DF 146          C=DAT0 A            Read the 2 bytes of name
1646 F18E2 96A          ?C=0   B            Zero byte?
1647 F18E5 83           GOYES  LIST50       Yes...done with list
1648 F18E7 145          DAT1=C A            No...write the bytes out
1649 F18EA 171          D1=D1+ 2
1650 F18ED F6           CSR    A            Check if second char was null
1651 F18EF F6           CSR    A            Now second char in C[B], C[4:2]=0
1652 F18F1 96E          ?C#0   B            Was the second char null?
1653 F18F4 90           GOYES  LIST30       No...continue
1654 F18F6 3102         LCASC  \ \
1655 F18FA 14D          DAT1=C B            Yes...replace with a blank
1656 F18FD 171  LIST30  D1=D1+ 2            Now D1 @ end of string
1657            *****
1658            *
1659            *       LC(8)  #FF0A0D*256+\'\  "'"&CR&LF&CHR$(255)
1660 F1900 37           NIBHEX 37
1661 F1902 72D0         NIBHEX 72D0A0FF
            A0FF
1662            *
1663            *****
1664 F190A 15D7         DAT1=C 8            Write it out
1665 F190E 1D00         D1=(2) =FUNCR0      Point back to start...
1666 F1912 8F00         GOSBVL =BF2DSP      ...and send to the display
            000
1667 F1919 6C5F         GOTO   LIST20       Loop back (not done yet)
1668            *-
1669            *-
1670 F191D      LIST50
```

```
1671            *
1672            * Done with LIST IO
1673            *
1674 F191D 626E          GOTO   nXTSTM
1675            ****************************************************************
1676            ****************************************************************
1677            **
1678            ** Name:        OFFIO - Execute the OFF IO statement
1679            **
1680            ** Category:    STEXEC
1681            **
1682            ** Purpose:
1683            **       Execute the "OFF IO" statement
1684            **
1685            ** Entry:
1686            **       Hexmode, P=0
1687            **
1688            ** Exit:
1689            **       Through NXTSTM
1690            **
1691            ** Calls:       D1=DST,<NXTSTM>
1692            **
1693            ** Uses.......
1694            **  Inclusive: A[B],C[A],D0,D1
1695            **
1696            ** Stk lvls:   0
1697            **
1698            ** History:
1699            **
1700            **    Date      Programmer              Modification
1701            **  --------    ----------    ---------------------------------
1702            **  12/15/82       NZ         Updated documentation
1703            **
1704            ****************************************************************
1705            ****************************************************************
1706 F1921 0000          REL(5) =OFFIOd        Decompile "IO"
           0
1707 F1926 0000          REL(5) =OFFIOp        Parse OFF IO/INTR
           0
1708 F192B 14A  =OFFIO   A=DAT0 B              Read the first token to check
1709 F192E 3100          LC(2)  =tXWORD          for IO vs INTR
1710 F1932 966           ?A#C   B              Is it INTR?
1711 F1935 11            GOYES  OFFIO1         No...must be OFF IO
1712            *
1713            * It is OFF INTR; clear the ONINTR address
1714            *
1715 F1937 1F00          D1=(5) =ONINTR
           000
1716 F193E D2            C=0    A
1717 F1940 145           DAT1=C A
1718 F1943 5D2           GONC   OFFIO2         Go always
1719            *-
1720            *-
1721 F1946 1F00 OFFIO1   D1=(5) =LOOPST
           000
```

```
1722 F194D 1572          C=DAT1 XS
1723 F1951 0B            CSTEX
1724 F1953 850           ST=1   =Offed        Loop is OFFED by the user
1725 F1956 0B            CSTEX
1726 F1958 1552          DAT1=C XS            Write it back out
1727               *
1728 F195C 8E00          GOSUBL =D1=DST
          00
1729 F1962 1572          C=DAT1 XS
1730 F1966 0B            CSTEX
1731 F1968 840           ST=0   =LoopOK       Loop is NOT ok
1732 F196B 0B            CSTEX
1733 F196D 1552          DAT1=C XS
1734 F1971 6E0E OFFIO2   GOTO   nXTSTM        Exit through NXTSTM
1735               ****************************************************************
1736               ****************************************************************
1737               **
1738               ** Name:       RESTIO - Execute the RESTORE IO statement
1739               ** Name:       REST1O - RESTORE IO, loop # in C[S]
1740               **
1741               ** Category:   STEXEC
1742               **
1743               ** Purpose:
1744               **      Execute the RESTORE IO statement...undo the effects
1745               **      of an OFF IO and reinitialize the specified loop
1746               **
1747               ** Entry:
1748               **      HEXMODE, P=0
1749               **
1750               ** Exit:
1751               **      Through ENDST if no error, through ERRORX if error
1752               **
1753               ** Calls:      CKLOP#,D1=DST,START-,RESTRT,PILCNF,<ENDST>,
1754               **             <ERRORX>
1755               **
1756               ** Uses.......
1757               **  Inclusive: All CPU registers,ST[11:0],FUNCxx,all RAM that
1758               **             EXPEXC is permitted to use
1759               **
1760               ** Stk lvls:   7 (CKLOP#)
1761               **
1762               ** History:
1763               **
1764               **    Date     Programmer          Modification
1765               **  --------   ----------   --------------------------------
1766               **  08/12/83      NZ        Reordered code between RESTIO and
1767               **                          (former) REST1O to allow REST1O
1768               **                          to clear the OFFED flag
1769               **  08/05/83      NZ        Changed to take a loop number
1770               **  12/15/82      NZ        Updated documentation
1771               **
1772               ****************************************************************
1773               ****************************************************************
1774 F1975 0000          REL(5) =RESTd
          0
```

```
1775 F197A 0000          REL(5) =RESTp
           0
1776 F197F 8E00 =RESTIO GOSUBL =CKLOP#       Get loop number, if any
           00
1777               *
1778               * C[S] is the loop number
1779               *
1780               * (Entry for ASSIGN IO "" and CONTROL ON)
1781               *
1782 F1985 1F00 =REST10 D1=(5) =LOOPST
           000
1783 F198C D2            C=0    A             Clear all bits in nibble
1784 F198E 15D0          DAT1=C 1             Loop is no longer offed
1785               *
1786               * Set the loop OK flag for the display device
1787               *
1788 F1992 8E00          GOSUBL =D1=DST
           00
1789 F1998 1572          C=DAT1 XS
1790 F199C 0B            CSTEX
1791 F199E 850           ST=1   =LoopOK       Set the loop "OK"
1792 F19A1 0B            CSTEX
1793 F19A3 1552          DAT1=C XS            Write it back out to RAM
1794               *
1795               * Now readdress loop (loop # still in C[S])
1796               *
1797 F19A7 850           ST=1   =sReadd       Force readdressing
1798 F19AA D3            D=0    A             Set device = NULL
1799               *
1800               * With device=null, START- will not error out if that loop
1801               * is currently in device mode, but will just return
1802               *
1803 F19AC 8E00          GOSUBL =START-       Readdress the loop if controller
           00
1804 F19B2 4C0           GOC    Rester        Error during START
1805 F19B5 8E00          GOSUBL =PILCNF       Restore OFFED devices, set DSPCHX
           00
1806 F19BB 648C          GOTO   Endst         Done...exit
1807               *-
1808               *-
1809 F19BF 654C Rester   GOTO   Errorx        Error jump
1810               ********************************************************************
1811               ********************************************************************
1812               **
1813               ** Name:       ASGNIO - Execute the ASSIGN IO statement
1814               **
1815               ** Category:   STEXEC
1816               **
1817               ** Purpose:
1818               **      Execute the ASSIGN IO statement (undo all DISPLAY IS
1819               **      and PRINTER IS assignments, allocate/deallocate the
1820               **      assign io device buffer
1821               **
1822               ** Entry:
1823               **      D0 points to the device specifier list
```

```
1824                **      P=0, HEXMODE
1825                **
1826                ** Exit:
1827                **      Through ENDST if no error, ERRORX if error
1828                **
1829                ** Calls:      GETSTR,TSAVD0,TSAVD1,NXTCHR,I/ODAL,D1=DSP,I/OALL,
1830                **              START-,TRESD0,TSWAD1,UCRANG,ASRC2,CATCH+,BAKCHR,
1831                **              ASLC2,<REST10>,<BSERR>,<ENDST>
1832                **
1833                ** Uses.......
1834                **  Inclusive: All CPU registers,ST[11:0],STMTD0,STMTR1,FUNCxx,
1835                **              all RAM EXPEXC is permitted to use
1836                **
1837                ** Stk lvls:   7 (GETSTR)
1838                **
1839                ** History:
1840                **
1841                **    Date      Programmer              Modification
1842                **  --------    ----------    ---------------------------------
1843                **  11/30/83      NZ          Updated documentation
1844                **  12/21/82      NZ          Added documentation
1845                **
1846                ****************************************************************
1847                ****************************************************************
1848 F19C3 0000          REL(5) =ASGNd
                0
1849 F19C8 0000          REL(5) =ASGNp
                0
1850 F19CD        =ASGNIO
1851                *
1852                * Get the string from program memory
1853                *
1854 F19CD 8E00          GOSUBL =GETSTR
                00
1855                *
1856                * GETSTR returns two cases:
1857                *    1) (Literal expression): ST(=sSTK)=0, D0 at start of data
1858                *    2) (String expression): ST(=sSTK)=1, D1 at start of data,
1859                *                            D[A] past end of data
1860                *
1861                * If ST(=sSTK)=0, then this is ASSIGN IO *
1862                *
1863 F19D3 860           ?ST=0  =sSTK          Reading from stack?
1864 F19D6 81            GOYES  ASGNOO         No...ASSIGN IO *
1865                *
1866                * Reading from stack (ASSIGN IO "????")
1867                *
1868 F19D8 7DBC          GOSUB  TsavdO         Save D0 (to restore after I/OALL)
1869 F19DC 8E00          GOSUBL =TSAVD1        Save D1
                00
1870 F19E2 DB            C=D    A
1871 F19E4 108           RO=C                  Save end (if string) in RO
1872                *
1873                * The exit conditions of GETSTR match those needed be NXTCHR!
1874                *
```

```
1875 F19E7 7C7C          GOSUB  Nxtchr        Check if this is a "*"
1876 F19EB 5D0           GONC   ASGN04        No error...exit
1877              *
1878              * ASSIGNIO "" = deallocate the ASSIGNIO buffer
1879              *
1880 F19EE 7631 ASGN00   GOSUB  ASGNda        Deallocate,
1881 F19F2 AC2           C=0    S                (loop 1!)
1882 F19F5 6F8F          GOTO   REST10           exit through restore
1883              *-
1884              *-
1885 F19F9       ASGN04
1886 F19F9 31A2          LCASC  \*\
1887 F19FD 962           ?A=C   B
1888 F1A00 EE            GOYES  ASGN00
1889              *
1890              * Not "*"...Unassign all devices
1891              *
1892              * (ASSIGN IO "device list")
1893              *
1894 F1A02 8E00          GOSUBL =D1=DSP
          00
1895 F1A08 AF2           C=0    W             C[W]="000...000"
1896 F1A0B A7E           C=C-1  W             C[W]="FFF...FFF"
1897 F1A0E 15DD          DAT1=C 14            Clear IS-DSP, IS-PRT
1898 F1A12 17D           D1=D1+ 14
1899 F1A15 15DD          DAT1=C 14            Clear IS-INP, IS-PLT
1900              *
1901              * Now create the I/O buffer for the ASSIGN words
1902              *
1903 F1A19 D2            C=0    A
1904              *
1905              * Leave 1 byte @ end (terminates LISTIO)
1906              *
1907 F1A1B 31A7          LC(2)  30*2*2+1*2    30 entries of 2 bytes, 2 nib/byte
1908 F1A1F D5            B=C    A             Size in B[A]
1909 F1A21 3200          LC(3)  =bPILAI       Assign IO
          0
1910 F1A26 8F00          GOSBVL =I/OALL       I/O ALLocate routine
          000
1911 F1A2D 490           GOC    ASGN05        OK
1912 F1A30 8D00 =bSERR   GOVLNG =BSERR        Error (mem)
          000
1913              *-
1914              *-
1915              *
1916              * The I/O buffer is allocated, D1 is the start of the buffer
1917              *
1918              * Initialize the buffer to all zero
1919              *
1920 F1A37 137   ASGN05   CD1EX                Get D1 value into D0...
1921 F1A3A 135           D1=C                 ...restore D1
1922 F1A3D 134           D0=C                 Use D0 for clear loop
1923 F1A40 AF2           C=0    W
1924 F1A43 28            P=     16-(120/15)   (30*2*2 = 120)
1925 F1A45 15CE ASGN10   DAT0=C 15
```

```
1926 F1A49 16E          D0=D0+ 15
1927 F1A4C 0C           P=P+1
1928 F1A4E 56F          GONC   ASGN10        Loop back if not done yet
1929 F1A51 14C          DAT0=C B             Clear out terminator byte
1930               *
1931               * Now D1 points to the buffer area, A[A] is length
1932               * FUNCD0 contains the program pointer
1933               *
1934               * Set OFFED flag = 0 (ASSIGN IO eliminates OFF IO)
1935               *
1936 F1A54 1B00         D0=(5) =LOOPST
           000
1937 F1A5B D2           C=0    A
1938 F1A5D 1542         DAT0=C XS            No longer OFFED, no devices set up
1939               *
1940               * Now readdress the loop (Primary only), use last address as
1941               * a device count
1942               *
1943 F1A61 AC2          C=0    S             Always loop 1 for ASSIGN IO
1944               *
1945               * Since D[A] is the end of the string and could look like
1946               * a request to search for the device to START-, set D[0] to
1947               * 1 (which always looks like an address, no search).  This also
1948               * ensures that the HP-71 is the controller on loop 1.
1949               *
1950 F1A64 BF3          DSL    W
1951 F1A67 E7           D=D+1  A             D[0] is now "1"
1952 F1A69 850          ST=1   =sReadd       Force readdressing
1953 F1A6C 8E00         GOSUBL =START-       Set it up (first mailbox)
           00
1954 F1A72 560          GONC   ASGN15        Found it, controller...ok
1955 F1A75 6E90         GOTO   ASGNeR        Not found or not controller...error
1956               *-
1957               *-
1958               *
1959               * If start returns with no carry, then last message in MBOX is
1960               * the address message from readdressing the loop
1961               *
1962 F1A79 BF7  ASGN15  DSR    W             First restore D[A]
1963 F1A7C 169          D0=D0+ 10            Position to the message in mailbox
1964 F1A7F 14E          C=DAT0 B             Read address
1965 F1A82 189          D0=D0- 10            Restore D0
1966               *
1967               * Now C[B] is the last address
1968               *
1969 F1A85 D5           B=C    A             Save count in B[B]
1970 F1A87 CD           B=B-1  A             Decrement for zero-based loop
1971               *
1972 F1A89 721C         GOSUB  TresD0        Restore D0
1973 F1A8D 8E00         GOSUBL =TSWAD1       Restore D1, save buffer pointer
           00
1974 F1A93 118          C=R0
1975 F1A96 D7           D=C    A             Restore end of string pointer
1976               *
1977               * Now D1 is restored, buffer pointer is in FUNCD1
```

```
1978                *
1979                * Loop to get ASSIGN words, copy to assign buffer
1980                *
1981 F1A98 7BCB ASGN20  GOSUB  Nxtchr        Get first character
1982 F1A9C 560         GONC   ASGN30        Have NOT reached end of string
1983 F1A9F 60AB ASGN25  GOTO   Endst         Reached end (Done)
1984                *-
1985                *-
1986 F1AA3 8E00 ASGN30  GOSUBL =UCRANG       Check if a letter (convert to UC)
           00
1987 F1AA9 5B1         GONC   ASGN40        Now in [A-Z]...continue
1988 F1AAC 31A3        LCASC  \:\           Not in [A-Z]...check if ":"
1989 F1AB0 966         ?A#C   B             Is it a ":"?
1990 F1AB3 F5          GOYES  ASGNER        No...bad character error
1991 F1AB5 7EAB        GOSUB  Nxtchr        Get next character
1992 F1AB9 485         GOC    ASGNER        End of list after ":"...error
1993 F1ABC 8E00        GOSUBL =UCRANG       Check if a letter (convert to UC)
           00
1994 F1AC2 4F4         GOC    ASGNER        Not in [A-Z]...error
1995                *
1996                * Letter...save in A[15:14]
1997                *
1998 F1AC5 8E00 ASGN40  GOSUBL =ASRC2
           00
1999 F1ACB 789B        GOSUB  Nxtchr        Read next character
2000 F1ACF 411         GOC    ASGN45        End of string...single letter word
2001 F1AD2 8E00        GOSUBL =cATCH+       Check if letter or digit
           00
2002 F1AD8 4A0         GOC    ASGN50        Letter or digit...OK
2003 F1ADB 8E00        GOSUBL =BAKCHR       Back up unconditionally
           00
2004 F1AE1 D0   ASGN45  A=0    A             Clear A[B] (single letter word)
2005                *
2006                * Valid word...save it in buffer
2007                *
2008 F1AE3 8E00 ASGN50  GOSUBL =TSWAD1       Swap D1 with buffer pointer
           00
2009 F1AE9 8E00        GOSUBL =ASLC2        Rotate word back to A[3:0]
           00
2010 F1AEF 1593        DAT1=A 4             Write the word out to the buffer
2011 F1AF3 173         D1=D1+ 4             Increment pointer
2012 F1AF6 8E00        GOSUBL =TSWAD1       Swap D1 back
           00
2013                *
2014                * Ready for next character
2015                *
2016 F1AFC 776B        GOSUB  Nxtchr
2017 F1B00 4E9         GOC    ASGN25        No next character (done)
2018 F1B03 31C2        LCASC  \,\           Got a character...check it
2019 F1B07 966         ?A#C   B             Is it a comma?
2020 F1B0A 80          GOYES  ASGNER        No...error
2021 F1B0C A6D         B=B-1  B             Yes...devices left to get words?
2022 F1B0F 588         GONC   ASGN20        Yes...continue on
2023                *
2024                * Fall through to error (too many words)
```

```
2025           *
2026 F1B12 20  ASGNER  P=     =eDSPEC       Invalid Device Spec
2027 F1B14 80C1 ASGNeR  C=P    1             Save P in C[1]
2028 F1B18 06           RSTK=C               Save error (in C[B]) on RSTK
2029 F1B1A 7A00         GOSUB  ASGNda        Deallocate assignio buffer
2030 F1B1E 07           C=RSTK               Restore error from RSTK
2031 F1B20 80D1         P=C    1             Restore P from C[1]
2032 F1B24 60EA         GOTO   Errorx        Error exit
2033           *-
2034           *-
2035 F1B28 20  ASGNda  P=     0             Deallocate the ASSIGN buffer
2036 F1B2A 3200         LC(3)  =bPILAI
          0
2037 F1B2F 8D00 =I/odal GOVLNG =I/ODAL
          000
2038           ****************************************************************
2039           ****************************************************************
2040           **
2041           ** Name:       DEVID - Return the device ID of the device
2042           **
2043           ** Category:   FNEXEC
2044           **
2045           ** Purpose:
2046           **      Return the device ID of the device indicated by the
2047           **      device specifier passed as a parameter
2048           **
2049           ** Entry:
2050           **      P=0
2051           **      D1 points to the stack
2052           **      D0 points to the PC
2053           **
2054           ** Exit:
2055           **      P=0
2056           **      D1 points to the  stack (Device ID string)
2057           **      Returns through FNRTN1
2058           **      If device not found/doesn't respond, null string
2059           **      If bad device spec, error
2060           **
2061           ** Calls:      DEVPAR,GETID+,ENDFN,TRESD0,<FNRTN1>,<ERRORX>
2062           **
2063           ** Uses.......
2064           **  Inclusive: A,B,C,D,R0-R3,D1,P,FUNCD0,FUNCD1,MLFFLG,ST[7,4:0]
2065           **
2066           ** Stk lvls:   4 (DEVPAR)
2067           **
2068           ** History:
2069           **
2070           **    Date     Programmer            Modification
2071           **  --------   ----------    ---------------------------------
2072           **  09/07/83      NZ         Packed at DEVID3
2073           **  12/21/82      NZ         Updated documentation
2074           **
2075           ****************************************************************
2076           ****************************************************************
2077 F1B36 C11          NIBHEX C11           One parameter, string or numeric
```

```
2078 F1B39 7841 =DEVID  GOSUB  DEVPAR        Get parameter
2079 F1B3D 485          GOC    DEVIDe        Error
2080            *
2081            * Now D[A] is address of the device
2082            *
2083            * If D[A]=0, then not found...return null string
2084            *
2085 F1B40 AC3          D=0    S             D[S] = length of ID in characters
2086 F1B43 8AB          ?D=0   A             Found?
2087 F1B46 B0           GOYES  DEVID1        No...null ID
2088            *
2089            * Get the device ID of the device
2090            *
2091 F1B48 8E00         GOSUBL =GETID+       Get Device ID of device
          00
2092            *
2093            * GETID returns with the ID in A[W]. The length in characters
2094            * is in D[S]. A[B] is the first character of the ID.
2095            *
2096 F1B4E 474          GOC    DEVIDe        Error if carry
2097 F1B51      DEVID1
2098            *
2099            * Now D1 @ stack-16, D[S] is length of ID in nibbles, A[W] is
2100            * device ID of the device
2101            *
2102 F1B51 ACB          C=D    S
2103 F1B54 80DF         P=C    15            P is length in characters
2104 F1B58 17F          D1=D1+ 16            Point to top of stack (first item)
2105 F1B5B 890  DEVID2  ?P=    0             Is length zero yet?
2106 F1B5E 31           GOYES  DEVID3        Yes...done writing ID to stack
2107 F1B60 1C1          D1=D1- 2             No...write another byte
2108 F1B63 149          DAT1=A B
2109 F1B66 BF4          ASR    W
2110 F1B69 BF4          ASR    W             Set up next data item
2111 F1B6C 0D           P=P-1
2112 F1B6E 5CE          GONC   DEVID2        Go always (P was not zero)
2113            *-
2114            *-
2115 F1B71      DEVID3
2116            *
2117            * Now write out the string header
2118            *
2119 F1B71 A46          C=C+C  S             Convert to number of nibbles
2120 F1B74 80DF         P=C    15            Now P is number of nibbles
2121 F1B78 AF2          C=0    W             Clear C[W] for string header
2122 F1B7B 80F2         CPEX   2             String length in C[2], P=0
2123            *
2124            * If carry, then length=8...increment C[3] (C[M])
2125            *
2126 F1B7F 550          GONC   DEVID4
2127 F1B82 B56          C=C+1  M             C[3]=1
2128 F1B85 A0E  DEVID4  C=C-1  P             C[0]="F" (string header)
2129 F1B88 8E00         GOSUBL =ENDFN        Clean up loop (C saved in R0)
          00
2130 F1B8E 7D0B         GOSUB  TresD0        Restore D0 value (PC)
```

```
2131 F1B92 6CF1          GOTO   Fnrtn1        Return, C[W] is string header
2132              *-
2133              *-
2134 F1B96 6E6A DEVIDe   GOTO   Errorx         Error
2135              ****************************************************************
2136              ****************************************************************
2137              **
2138              ** Name:       SPOLL - Execute the SPOLL function
2139              **
2140              ** Category:   FNEXEC
2141              **
2142              ** Purpose:
2143              **      SPOLL is a function which returns the status of the
2144              **      device specified by either an address or a string
2145              **      device specifier.
2146              **
2147              ** Entry:
2148              **      P=0
2149              **      D0 points to PC
2150              **      D1 points to the top of the stack (device spec)
2151              **
2152              ** Exit:
2153              **      P=0
2154              **      Numeric value on stack (D1 points to top of stack),
2155              **        value = -1 if device not found or no response
2156              **        (the numeric value is the decimal equivalent of the
2157              **         first 4 bytes of device status...because more than
2158              **         four bytes may lose accuracy in the conversion to
2159              **         decimal; 2^(8*5) is about 1.1E+12, which would lose
2160              **         a small amount of precision in the FIRST byte.
2161              **         The first byte is SPOLL(x) mod 256, etc
2162              **      Returns through FNRTN4
2163              **      If error, exits through ERRORX
2164              **
2165              ** Calls:      DEVPAR,YTML,READRG,<DEVTYx>,<ERRORX>
2166              **
2167              ** Uses.......
2168              **  Inclusive: A,B,C,D,R0-R3,D1,P,FUNCD0,FUNCD1,MLFFLG,ST[7,4:0]
2169              **
2170              ** Stk lvls:   4 (DEVPAR)
2171              **
2172              **
2173              ** History:
2174              **
2175              **    Date     Programmer            Modification
2176              **  --------   ----------   --------------------------------
2177              **  03/15/83      NZ        Removed extra START call @ SPOL10
2178              **  02/25/83      NZ        Modified to change order of bytes
2179              **  02/24/83      SC        Wrote routine
2180              **
2181              ****************************************************************
2182              ****************************************************************
2183 F1B9A C11           NIBHEX C11           1 parameter, either numeric/string
2184 F1B9D 74E0 =SPOLL   GOSUB  DEVPAR        Process device specifier
2185 F1BA1 4D3           GOC    FINDer        Error
```

```
2186                *
2187                * D[X] is the device address (D[X]=0 if not found)
2188                *
2189 F1BA4 93F          ?D#0   X             Was the device found?
2190 F1BA7 60           GOYES  SPOL10        Yes...continue
2191                *
2192                * If device not found, return -1
2193                *
2194 F1BA9 65C0 SPOL05  GOTO   DEVTY5
2195                *-
2196                *-
2197 F1BAD 8E00 SPOL10  GOSUBL =YTML         Make the device as a talker
           00
2198 F1BB3 4B2          GOC    FINDer        Error
2199                *
2200                * Only the first 4 bytes are returned, but READRG expects 8
2201                *
2202 F1BB6 3500         LC(6)  (=MSST)+#8    Send ready frame SST, count=8
           0000
2203 F1BBE 8E00         GOSUBL =READRG       Read into A[W]
           00
2204 F1BC4 4A1          GOC    FINDer        Error
2205 F1BC7 94B          ?D=0   S             Any response?
2206 F1BCA FD           GOYES  SPOL05        No...return -1
2207 F1BCC AF2          C=0    W             Clear high 4 bytes
2208 F1BCF 27           P=     7
2209 F1BD1 A96          C=A    WP            Return only first 4 bytes
2210 F1BD4 6680         GOTO   DEVTYx        Convert to floating number, exit
2211                ****************************************************************
2212                ****************************************************************
2213                **
2214                ** Name:       FIND - Execute the DEVADDR function
2215                **
2216                ** Category:   FNEXEC
2217                **
2218                ** Purpose:
2219                **      FIND is a function which returns the address of the
2220                **      device specified by either an address (trival case) or
2221                **      a string device specifier
2222                **
2223                ** Entry:
2224                **      P=0
2225                **      D0 points to the PC
2226                **      D1 points to the stack (device specifier on stack)
2227                **
2228                ** Exit:
2229                **      P=0
2230                **      Numeric expression on stack (D1 points to the address)
2231                **         (-1=not found, else address)
2232                **      Returns through FNRTN4
2233                **      If error, exits through ERRORX
2234                **
2235                ** Calls:      DEVPAR,HTOD,CSLC12,<DEVTY4>,<DEVTY5>,<ERRORX>
2236                **
2237                ** Uses.......
```

```
2238                ** Inclusive: A,B,C,D,R0-R3,D1,P,FUNCD0,FUNCD1,MLFFLG,ST[7,4:0]
2239                **
2240                ** Stk lvls:   4 (DEVPAR)
2241                **
2242                ** History:
2243                **
2244                **    Date     Programmer            Modification
2245                **  --------   ----------   --------------------------------
2246                **  12/21/82      NZ        Updated documentation
2247                **
2248                ********************************************************************
2249                ********************************************************************
2250 F1BD8 C11          NIBHEX C11            One argument, string or numeric
2251 F1BDB 76A0 =FIND   GOSUB  DEVPAR         Evaluate the device specifier
2252 F1BDF 455  FINDer  GOC    FINDER         Error
2253                *
2254                * Convert D[X] to a floating number address
2255                *
2256 F1BE2 D2           C=0    A
2257 F1BE4 20           P=     0
2258 F1BE6 31F1         LC(2)  #1F            Get primary address
2259 F1BEA 0EF7         C=C&D  A
2260 F1BEE 8E00         GOSUBL =HTOD          Convert to decimal (in B[X])
          00
2261 F1BF4 D4           A=B    A              Save in A[X]
2262                *
2263                * Now A[X] is the primary address value
2264                *
2265 F1BF6 20           P=     0
2266 F1BF8 320E         LC(3)  #3E0           Mask for secondary address
          3
2267 F1BFD 0EF7         C=C&D  A              C[X] is secondary * 32
2268 F1C01 BB6          CSR    X              Cannot be CSR A:need C[XS]=xxx0(2)
2269 F1C04 81E          CSRB                  C[B] is secondary address
2270 F1C07 8E00         GOSUBL =HTOD          Convert to decimal
          00
2271                *
2272                * Now DECIMAL mode, B[X] is secondary address, A[X] is primary
2273                *
2274 F1C0D 04           SETHEX
2275 F1C0F 20           P=     0              HTOD leaves P non-zero
2276 F1C11 AF2          C=0    W
2277 F1C14 D6           C=A    A              Copy A[B] (A[4:2]=0)
2278 F1C16 F2           CSL    A
2279 F1C18 F2           CSL    A
2280 F1C1A AE9          C=B    B              Now C[3:2] is primary, [B] is sec
2281 F1C1D 8E00         GOSUBL =CSLC12        Rotate into C[15:12]
          00
2282                *
2283                * If C[S] is non-zero, shift RIGHT 1 nibble, add 1 to exponent
2284                * (address is >= 10)
2285                *
2286 F1C23 94A          ?C=0   S
2287 F1C26 70           GOYES  FIND10
2288 F1C28 BF6          CSR    W              (Exponent, low mantissa = 0)
```

```
2289 F1C2B E6          C=C+1  A            C[X]=1
2290 F1C2D      FIND10
2291            *
2292            * Now C[W] is value, D1 points to the stack
2293            *
2294 F1C2D 97E         ?C#0   W            Is it zero? (not found)
2295 F1C30 13          GOYES  DEVTY4       No...value is OK
2296 F1C32 5C3         GONC   DEVTY5       Yes...return -1 (not found)
2297            *-
2298            *-
2299 F1C35 6FC9 FINDER  GOTO   Errorx       Error
2300            ****************************************************************
2301            ****************************************************************
2302            **
2303            ** Name:       DEVTYP - Execute the DEVAID function
2304            **
2305            ** Category:   FNEXEC
2306            **
2307            ** Purpose:
2308            **      DEVTYP returns the accessory ID of the device indicated
2309            **      by the device specifier
2310            **
2311            ** Entry:
2312            **      P=0
2313            **      D1 points to the stack (device specifier on the stack)
2314            **      D0 points to the PC
2315            **
2316            ** Exit:
2317            **      P=0
2318            **      Numeric expression for accessory ID (-1 if no response)
2319            **      Returns through FNRTN4
2320            **      Exits through ERRORX if error
2321            **
2322            ** Calls:      DEVPAR,GTYPE,FLOAT!,ENDFN,TRESD0,<FNRTN4>,<ERRORX>
2323            **
2324            ** Uses.......
2325            **  Inclusive: A,B,C,D,R0-R3,D1,P,FUNCD0,FUNCD1,MLFFLG,ST[7,4:0]
2326            **
2327            ** Stk lvls:   4 (DEVPAR)
2328            **
2329            ** History:
2330            **
2331            **    Date     Programmer          Modification
2332            **  --------   ----------   ---------------------------------
2333            **  05/17/83      NZ        Changed return from GTYPE
2334            **  12/21/82      NZ        Added documentation
2335            **
2336            ****************************************************************
2337            ****************************************************************
2338 F1C39 C11           NIBHEX C11          One parameter, string or numeric
2339 F1C3C 7540 =DEVTYP  GOSUB  DEVPAR       Get device parameter
2340 F1C40 404           GOC    DEVTYe       Error
2341            *
2342            * Now D0 points to the mailbox, D[X] is the address
2343            *
```

```
2344 F1C43 8AB          ?D=0   A            Was the device found?
2345 F1C46 92           GOYES  DEVTY5       No...return -1
2346 F1C48 8E00         GOSUBL =GTYPE       Get device type for the device
           00
2347 F1C4E 423          GOC    DEVTYe       Error...exit
2348 F1C51 8A8          ?A=0   A            Was it "NO RESPONSE"?
2349 F1C54 B1           GOYES  DEVTY5       Yes...return -1
2350 F1C56 AF2          C=0    W
2351 F1C59 D6           C=A    A            Copy all info returned from GTYPE
2352 F1C5B 8E00 DEVTYx  GOSUBL =FLOAT!      Convert to floating point #
           00
2353 F1C61 8E00 DEVTY4  GOSUBL =ENDFN       Clean up the loop
           00
2354 F1C67 743A         GOSUB  TresdO       Restore DO
2355 F1C6B 6F02         GOTO   Fnrtn4       Return the value
2356            *-
2357            *-
2358 F1C6F 7300 DEVTY5  GOSUB  LOAD-1       Load a -1 into C[W]
2359 F1C73 5DE          GONC   DEVTY4       Go always
2360            *-
2361            *-
2362 F1C76 AF2  LOAD-1  C=0    W
2363 F1C79 2E           P=     14
2364 F1C7B 3119         LCHEX  91           This is -1
2365 F1C7F 03           RTNCC
2366            *-
2367            *-
2368 F1C81 6389 DEVTYe  GOTO   Errorx       Error
2369            ****************************************************************
2370            ****************************************************************
2371            **
2372            ** Name:       DEVPAR - Parse a device specifier on the stack
2373            ** Name:       DEVPR$ - Parse a string device spec on stack
2374            **
2375            ** Category:   PILUTL
2376            **
2377            ** Purpose:
2378            **      Decode a device parameter (for functions which accept
2379            **      one parameter, either string or numeric, for device
2380            **      specifier)
2381            **
2382            ** Entry:
2383            **      P=0
2384            **      HEXMODE
2385            **      DEVPAR:
2386            **         D1 points to the parameter on stack
2387            **      DEVPR$:
2388            **         D1 points to string header (String is reversed)
2389            **         ST(sSTK)=1
2390            **
2391            ** Exit:
2392            **      FUNCDO contains the calling routine's DO value
2393            **      Carry clear: OK...D[X] is address (0 if not found)
2394            **             D1 set up for 1 numeric parameter return
2395            **             DO points to the mailbox
```

```
2396            **      Carry set: Error...P, C[0] set up for ERRORX
2397            **
2398            ** Calls:      TSAVD0,POP1N,GADRRM,REVPOP,<DEVPR$>
2399            **      DEVPR$:TSAVD1,GETDIX,TRESD1
2400            **
2401            ** Uses.......
2402            **  Inclusive: A,B,C,D,R0-R3,D1,P,FUNCD0,FUNCD1,MLFFLG,ST[7,4:0]
2403            **
2404            ** Stk lvls:   3 (GETDIX - two levels saved in R0)
2405            **
2406            ** History:
2407            **
2408            **    Date     Programmer          Modification
2409            **  --------   ----------   --------------------------------
2410            **  01/06/84      NZ        Made setting of MLFFLG a GOSUB so
2411            **                          code can be shared by READxxxx and
2412            **                          STATUS; moved call to the routine
2413            **                          so that DEVPR$ also sets MLFFLG
2414            **  03/16/83      NZ        Changed error return from GETDIX
2415            **  03/15/83      NZ        Added second stack level save for
2416            **                          call to GETDIX
2417            **  12/21/82      NZ        Updated documentation
2418            **
2419            ****************************************************************
2420            ****************************************************************
2421 F1C85    =DEVPAR
2422 F1C85 1537         A=DAT1 W            Read in the item from the stack
2423 F1C89 850          ST=1   =sSTK        GADDRM needs this if not a string
2424 F1C8C B04          A=A+1  P
2425 F1C8F A64          A=A+A  B            Clear bit for string array
2426 F1C92 968          ?A=0   B            Is this a string?
2427 F1C95 F2           GOYES  DEVP10       Yes...string device spec
2428            *
2429            * Not string...check for legal input
2430            *
2431 F1C97 7EF9         GOSUB  TsavD0       Save D0 in FUNCD0 (exit condition)
2432 F1C9B 78EA         GOSUB  Pop1n        Pop one numeric item into A[W]
2433            *
2434            * Now A[W] is the numeric item
2435            *
2436 F1C9F 8E00         GOSUBL =GADRRM      Get address from RAM (use A[W])
         00
2437 F1CA5 D7           D=C    A            Put address into D[A]
2438            *
2439            * If carry clear, C[X] is address else error
2440            *
2441 F1CA7 6060         GOTO   DEVP20       Check error, continue, C[A] is addr
2442            *-
2443            *-
2444 F1CAB 0            CON(1) =FIXSPC      3 nibbles available here
2445 F1CAC              BSS    3-1
2446            *-
2447            *-
2448            *
2449            * Set the MLFFLG to "F" (Sets A[A] to D0 value, C[0] to "F")
```

```
2450                *
2451 F1CAE 132     MLFG=F  ADOEX               Save DO in A[A]
2452 F1CB1 1B00            DO=(5) =MLFFLG
          000
2453 F1CB8 30F             LC(1)  #F           Set C[0]="F"
2454 F1CBB 15C0            DATO=C 1            Write it out
2455 F1CBF 130             DO=A                Restore DO from A[A]
2456 F1CC2 01              RTN
2457                *-
2458                *-
2459 F1CC4         DEVP10
2460                *
2461                * String item...set it up, call GETDIX with ST(sSTK)=1
2462                *
2463 F1CC4 8F00            GOSBVL =REVPOP      Reverse the string, POP it
          000
2464 F1CCB 137     =DEVPR$ CD1EX
2465 F1CCE D7              D=C    A            D points to start of string
2466 F1CD0 C2              C=C+A  A            C[A] points to end of string
2467 F1CD2 135             D1=C                Now D1, C[A] at end of string
2468 F1CD5 1CF             D1=D1- 16           Point to where numeric should go
2469 F1CD8 8E00            GOSUBL =TSAVD1      Save in FUNCD1
          00
2470 F1CDE DF              CDEX   A            D[A] at end of string,C[A] at start
2471 F1CE0 135             D1=C                Set D1 to the start of string
2472                *
2473                * Now D[A], D1 are set up for a call to GETDIX (Later entry
2474                * into GETDID)
2475                *
2476 F1CE3 07              C=RSTK
2477 F1CE5 7A89            GOSUB  Cslc5
2478 F1CE9 07              C=RSTK
2479 F1CEB 108             R0=C                Save 2 RSTK levels in R0
2480                *
2481                * GETDIX saves DO in FUNCDO...
2482                *
2483 F1CEE 8E00            GOSUBL =GETDIX      Get device address (in D[X])
          00
2484                *
2485 F1CF4 128             CROEX               Save C[W] in R0...
2486 F1CF7 06              RSTK=C              Restore first level...
2487 F1CF9 7D79            GOSUB  Csrc5
2488 F1CFD 06              RSTK=C              Restore second level
2489 F1CFF 118             C=R0                Restore C[A] value
2490                *
2491                * Now restore D1 value for exit, then check error
2492                *
2493 F1D02 8ECD            GOSUBL Tresd1       D1 @ next item on stack
          2F
2494                *
2495                * D1 is now where next item goes, C[A] is address,B[W] is type
2496                *
2497                * If carry, had an error (GETDIX did START)
2498                *
2499 F1D08 461     DEVP20  GOC    DEVP25       Go if error
```

```
2500 F1D0B 7F9F          GOSUB  MLFG=F        Set MLFFLG to "F"
2501 F1D0F 8E00          GOSUBL =START        (START for DEVP20 entry)
           00
2502 F1D15 490  DEVP23   GOC    DEVP25        Error...check what it is
2503 F1D18 96F           ?D#0   B             Is this a valid device spec?
2504 F1D1B 41            GOYES  DEVPcc        Yes...return, carry clear
2505               *
2506               * (Test at DEVP25 will be true, hence RTNSC...packing technique)
2507               *
2508 F1D1D 20            P=     =eDSPEC       No...Invalid Device Spec
2509               *
2510               * Error...check if "NOT FOUND" or something else
2511               *
2512 F1D1F 880  DEVP25   ?P#    =ePIL         PIL error?
2513 F1D22 00            RTNYES               No...some other error
2514 F1D24 80F0          CPEX   0
2515 F1D28 880           ?P#    =eNOFND       NOT FOUND?
2516 F1D2B 60            GOYES  DEVP30        (Set carry if not found)
2517               *
2518               * Error was "Device not Found"...set D[A]=0, continue
2519               *
2520 F1D2D D3            D=0    A
2521 F1D2F 03   DEVPcc   RTNCC
2522               *-
2523               *-
2524 F1D31 80F0 DEVP30   CPEX   0             Restore C[0],P
2525 F1D35 02            RTNSC                Set carry = error
2526               *-
2527               *-
2528 F1D37 0             CON(1) =FIXSPC       1 nibble available here
2529 F1D38               BSS    1-1
2530               ******************************************************************
2531               ******************************************************************
2532               **
2533               ** Name:       READIN - Execute the READ INTR function
2534               **
2535               ** Category:   FNEXEC
2536               **
2537               ** Purpose:
2538               **      Read the interrupt cause byte for the specified loop
2539               **      and return the value as a decimal number
2540               **
2541               ** Entry:
2542               **      P=0
2543               **      D1 points to the stack
2544               **      C[S]=number of parameters supplied by user
2545               **      If C[S]=1 then top of stack contains a numeric value
2546               **
2547               ** Exit:
2548               **      Numeric result on top of stack
2549               **      D1 at top of stack
2550               **      P=0
2551               **      Returns through FNRTN4
2552               **
2553               ** Calls:      GETLPs,PUTGF-,LOAD-1,FLOAT!,TRESD0,<FNRTN1>
```

```
2554                **
2555                ** Uses.......
2556                **  Inclusive: A,B,C,D,R0,D1,P,FUNCD0,ST[5,3:0]
2557                **
2558                ** Stk lvls:   3 (GETLPs)
2559                **
2560                ** History:
2561                **
2562                **    Date      Programmer            Modification
2563                **  --------    ----------    --------------------------------
2564                **  12/01/83       NZ         Updated documentation
2565                **  08/03/83       NZ         Added optional loop # (sharing
2566                **                            code with STATUS)
2567                **  05/20/83       NZ         Changed to save message in B[A]
2568                **                            instead of A[A] thru FNDMB-
2569                **  02/28/83       NZ         Changed to use TSAVD0 & TRESD0
2570                **                            instead of SAVED0 & RESTD0
2571                **                            Reworked routine to reduce code
2572                **  02/07/83       SC         Wrote routine
2573                **
2574                ********************************************************************
2575                ********************************************************************
2576 F1D38 20    R&CVEu  P=     0
2577 F1D3A 300          LC(1)  =eUNEXP       Unexpected frame
2578 F1D3D 20           P=     =ePIL
2579 F1D3F 65C8 R&CVER  GOTO   Errorx
2580                *-
2581                *-
2582 F1D43 801          NIBHEX 801           Zero or one numeric parameter
2583 F1D46 7060 =READIN GOSUB  GETLPs         Get (optional) loop # from stack
2584 F1D4A 44F          GOC    R&CVER        Error with loop
2585 F1D4D 3100         LC(2)  =mREADI
2586 F1D51 845          ST=0   5
2587 F1D54 8E00 RD&CVT  GOSUBL =PUTGF-       Read the byte from mailbox
          00
2588 F1D5A 44E          GOC    R&CVER
2589 F1D5D 880          ?P#    =pDIAGL       Contents of location?
2590 F1D60 8D           GOYES  R&CVEu        No...unexpected frame
2591 F1D62 20           P=     0
2592 F1D64 DA           A=C    A             Yes...save in A[B]
2593 F1D66 865          ?ST=0  5             Read interrupt cause?
2594 F1D69 61           GOYES  FCNRT1        Yes...return all 8 bits
2595                *
2596                * READDDC...if zero, return -1, else return top 6 bits
2597                *
2598 F1D6B 96C          ?A#0   B             Any DDCs received?
2599 F1D6E 90           GOYES  R&CV10        Yes...return 6 bits
2600 F1D70 720F Fnrtn-  GOSUB  LOAD-1        No...return -1
2601 F1D74 561          GONC   Fnrtn.        Go always
2602                *-
2603                *-
2604 F1D77 31F3 R&CV10  LCHEX  3F            Only 6 bits for DDC
2605 F1D7B 0EF6         A=A&C  A
2606 F1D7F AF2  FCNRT1  C=0    W
2607 F1D82 AE6          C=A    B             Copy A[B] for conversion
```

```
2608 F1D85 8E00          GOSUBL =FLOAT!
           00
2609 F1D8B 7019 Fnrtn.   GOSUB  TresdO        Restore PC from FUNCDO
2610 F1D8F 8D00 Fnrtn1   GOVLNG =FNRTN1       Exit with memory check
           000
2611               ****************************************************************
2612               ****************************************************************
2613               **
2614               ** Name:       READDC - Execute the READDDC functino
2615               **
2616               ** Category:   FNEXEC
2617               **
2618               ** Purpose:
2619               **      Return the last device dependent command received (low
2620               **      6 bits of the DDT or DDL frame) as a decimal number
2621               **
2622               ** Entry:
2623               **      D1 points to the stack
2624               **      C[S] is the number of parameters passed to the function
2625               **      If C[S]=1, there is a numeric expression on top of stack
2626               **      P=0
2627               **
2628               ** Exit:
2629               **      D1 points to the top of the stack
2630               **      P=0
2631               **      Returns through FNRTN1
2632               **
2633               ** Calls:      GETLPs,<RD&CVT>
2634               **
2635               ** Uses.......
2636               **  Inclusive: A,B,C,D,R0,D1,P,FUNCD0,ST[5,3:0]
2637               **
2638               ** Stk lvls:   3 (GETLPs)
2639               **
2640               ** History:
2641               **
2642               **    Date     Programmer             Modification
2643               **  --------   ----------   --------------------------------
2644               **  08/03/83      NZ        Modified to take a loop #
2645               **  02/28/83      NZ        Updated documentation
2646               **  02/07/83      SC        Wrote routine
2647               **
2648               ****************************************************************
2649               ****************************************************************
2650 F1D96 801           NIBHEX 801            Zero or one numeric parameter
2651 F1D99 7D00 =READDC  GOSUB  GETLPs         Get (optional) loop # from stack
2652 F1D9D 41A           GOC    R&CVER         Error with loop specifier
2653 F1DA0 3100          LC(2)  =mREADC        Read last ddc
2654 F1DA4 855           ST=1   5
2655 F1DA7 5CA           GONC   RD&CVT         Go always
2656               ****************************************************************
2657               ****************************************************************
2658               **
2659               ** Name:       GETLPs - Get (optional) loop #, check status
2660               **
```

```
2661                ** Category:   PILUTL
2662                **
2663                ** Purpose:
2664                **      Check if a loop number was passed to a function; if
2665                **      so, get that mailbox, else get first mailbox.
2666                **      Check the status of the mailbox (reset?, etc)
2667                **
2668                ** Entry:
2669                **      P=0
2670                **      D1 points to the top of the stack
2671                **      C[S] is the parameter count (0 or 1)
2672                **      If C[S]=1, there is a numeric value on top of the stack
2673                **
2674                ** Exit:
2675                **      Carry clear:
2676                **        P=0
2677                **        D0 points to the mailbox
2678                **        Mailbox status in C[X]
2679                **        D1 at (new) top of stack (loop number is popped off)
2680                **        FUNCD0 contains the caller's D0
2681                **      Carry set:
2682                **        Error (P, C[0] are the error code)
2683                **
2684                ** Calls:      TSAVD0,POP1N,GHEXB+,<FNDCHK>
2685                **
2686                ** Uses.......
2687                **  Inclusive: A,B,C,D,R0,D0,D1,P,FUNCD0,ST[3:0]
2688                **
2689                ** Stk lvls:   2 (TSAVD0)(GHEXB+)(<FNDCHK>)
2690                **
2691                ** History:
2692                **
2693                **    Date     Programmer             Modification
2694                **  --------   ----------    ---------------------------------
2695                **  12/01/83      NZ         Added documentation
2696                **
2697                ****************************************************************
2698                ****************************************************************
2699 F1DAA 700F =GETLPs GOSUB  MLFG=F          Set MLFFLG to indicate loop changed
2700 F1DAE 77E8         GOSUB  TsavD0         Save PC in FUNCD0
2701 F1DB2 94A          ?C=0   S              Loop number specified?
2702 F1DB5 B2           GOYES  Fndchk         No...use default (=first loop)
2703 F1DB7 7CC9         GOSUB  Pop1n          Yes...get value from stack
2704 F1DBB 4A2          GOC    GETLPe         If complex number, error
2705 F1DBE 8E00         GOSUBL =GHEXB+        Convert value into HEX byte
          00
2706 F1DC4 432          GOC    ErrorX         Error
2707                *
2708                * B[B] is now the value of the expression (B[4:2]=0)
2709                *
2710 F1DC7 D2           C=0    A
2711 F1DC9 302          LC(1)  2              Max of 3 loops (0,1, or 2)
2712 F1DCC DD           BCEX   A              Loop number in C[A]
2713 F1DCE 20           P=     =eRANGE
2714 F1DD0 CE           C=C-1  A              Convert user input to base zero
```

```
2715 F1DD2 451          GOC    ErrorX
2716 F1DD5 8B5          ?B<C   A             Is the loop number less than 2?
2717 F1DD8 01           GOYES  ErrorX        No...error
2718 F1DDA 816          CSRC                 C[S] is now loop number
2719 F1DDD 17F          D1=D1+ 16            Pop the numeric field off the stack
2720 F1DE0 8C00 Fndchk  GOLONG =FNDCHK       Find the mailbox (P can be non-0)
          00
2721            *-
2722            *-
2723 F1DE6 20   GETLPe  P=     =eNNUMR       Not numeric data
2724 F1DE8 6C18 ErrorX  GOTO   Errorx
2725            ****************************************************************
2726            ****************************************************************
2727            **
2728            ** Name:       STATUS - Execute the STATUS function
2729            **
2730            ** Category:   FNEXEC
2731            **
2732            ** Purpose:
2733            **      Return mailbox status as a numeric value
2734            **
2735            ** Entry:
2736            **      P=0
2737            **      D1 points to the top of the stack
2738            **      C[S]=Number of parameters passed to this function
2739            **      If C[S]=1, there is a numeric value on top of the stack
2740            **
2741            ** Exit:
2742            **      P=0
2743            **      D1 points to the top of the stack
2744            **      Numeric value for STATUS on top of the stack
2745            **      Returns through FNRTN1
2746            **
2747            ** Calls:      GETLPs,GETHSS,<FCNRT1>,<FNRTN->
2748            **
2749            ** Uses.......
2750            **  Inclusive: A,B,C,D,R0,D1,P,FUNCDO,ST[11:0]
2751            **
2752            ** Stk lvls:   3 (GETLPs)
2753            **
2754            ** History:
2755            **
2756            **    Date     Programmer            Modification
2757            **  --------   ----------   --------------------------------
2758            **  12/01/83      NZ        Updated documentation
2759            **  08/03/83      NZ        Changed first part to a subroutine
2760            **                          to do multiple loop READINTR/DDC
2761            **  06/16/83      NZ        Changed -1 return to pack code
2762            **  03/09/83      NZ        Changed where PC is saved to RAM
2763            **  03/07/83      NZ        Fixed bug with mailbox not found,
2764            **                          added call to GETERR to clear
2765            **                          error bit in mailbox
2766            **  02/28/83      NZ        Added check for insufficient mem,
2767            **                          mailbox out of range, packed,
2768            **                          updated documentation
```

```
2769            **  02/08/83      SC          Wrote routine
2770            **
2771            ********************************************************************
2772            ********************************************************************
2773 F1DEC 801          NIBHEX 801
2774 F1DEF 77BF =STATUS GOSUB  GETLPs          Get loop number, check status
2775            *
2776            * If carry clear, C[X] is the Diamond status
2777            *
2778 F1DF3 5B0          GONC   STAT10          All OK...continue STATUS execution
2779 F1DF6 880          ?P#    =eNMBOX         Is error "No mailbox"?
2780 F1DF9 FE           GOYES  ErrorX          No...error exit
2781 F1DFB 647F         GOTO   Fnrtn-          Yes...return -1, restore PC, exit
2782            *-
2783            *-
2784            sStanb  EQU    7
2785            sLA     EQU    6
2786            sCA     EQU    5
2787            sTA     EQU    4
2788            sSRQR   EQU    3
2789            sEar    EQU    2
2790            sRemot  EQU    1
2791            sLLout  EQU    0
2792            *
2793 F1DFF 08   STAT10  CLRST                  Initially clear all status bits
2794 F1E01 F2           CSL    A
2795 F1E03 F2           CSL    A               C[4:2] is the loop status now
2796 F1E05 C6           C=C+C  A               Bit 11: Local Lockout
2797 F1E07 550          GONC   STAT21          Clear
2798 F1E0A 850          ST=1   sLLout          Set Local Lockout bit
2799 F1E0D C6   STAT21  C=C+C  A               Bit 10: Remote
2800 F1E0F 550          GONC   STAT22          Clear
2801 F1E12 851          ST=1   sRemot          Set Remote bit
2802 F1E15 C6   STAT22  C=C+C  A               Bit 9: Manual mode      (ignored)
2803 F1E17 C6           C=C+C  A               Bit 8: Data available   (ignored)
2804 F1E19 C6           C=C+C  A               Bit 7: Controller Standby
2805 F1E1B 550          GONC   STAT23          Clear
2806 F1E1E 857          ST=1   sStanb          Set Controller Standby bit
2807 F1E21 C6   STAT23  C=C+C  A               Bit 6: EAR enabled
2808 F1E23 550          GONC   STAT24          Clear
2809 F1E26 852          ST=1   sEar            Set EAR enabled bit
2810 F1E29 C6   STAT24  C=C+C  A               Bit 5: Configured       (ignored)
2811 F1E2B C6           C=C+C  A               Bit 4: Interrupt pending(ignored)
2812 F1E2D C6           C=C+C  A               Bit 3: System Controller(ignored)
2813 F1E2F C6           C=C+C  A               Bit 2: Talker Active
2814 F1E31 550          GONC   STAT25          Clear
2815 F1E34 854          ST=1   sTA             Set Talker Active bit
2816 F1E37 C6   STAT25  C=C+C  A               Bit 1: Listener
2817 F1E39 550          GONC   STAT26          Clear
2818 F1E3C 856          ST=1   sLA             Set Listener bit
2819 F1E3F C6   STAT26  C=C+C  A               Bit 0: Controller Active
2820 F1E41 550          GONC   STAT27          Clear
2821 F1E44 855          ST=1   sCA             Set Controller Active bit
2822 F1E47 0B   STAT27  CSTEX                  C[B] is now the byte for STATUS
2823 F1E49 DA           A=C    A               Put STATUS into A[B]
```

```
2824 F1E4B 8E00          GOSUBL =GETHSS       Get handshake nibble from mailbox
           00
2825 F1E51 860           ?ST=0  =hsLPRQ       SRQ received on loop?
2826 F1E54 A0            GOYES  STAT30         No...leave the bit clear
2827 F1E56 3180          LC(2)  2^sSRQR        Yes...set the SRQR bit
2828 F1E5A 0EFE          A=A!C  A
2829 F1E5E 602F STAT30   GOTO   FCNRT1         Restore PC, convert to float&exit
2830                     ********************************************************************
2831                     ********************************************************************
2832                     **
2833                     ** Name:       BINAND - Execute the BINAND function
2834                     ** Name:       BINIOR - Execute the BINIOR function
2835                     ** Name:       BINEOR - Execute the BINEOR function
2836                     ** Name:       BINCMP - Execute the BINCMP function
2837                     ** Name:       BIT    - Execute the BIT function
2838                     **
2839                     ** Category:   FNEXEC
2840                     **
2841                     ** Purpose:
2842                     **       Binary functions:
2843                     **         BINAND: Return the binary AND of two numbers
2844                     **         BINIOR: Return the binary inclusive OR of two numbers
2845                     **         BINEOR: Return the binary exclusive OR of two numbers
2846                     **         BINCMP: Return the binary complement of a number
2847                     **         BIT: Return the value of a specific bit in a number
2848                     **
2849                     ** Entry:
2850                     **       P=0
2851                     **       D1 points to the top of the stack
2852                     **       Two values on top of the stack (only one for BINCMP)
2853                     **
2854                     ** Exit:
2855                     **       P=0
2856                     **       Returns through FNRTN4
2857                     **
2858                     ** Calls:      POP2DH,POP1N(BINCMP),FLOAT!,<FNRTN4>,<ERRORX>
2859                     **
2860                     ** Uses.......
2861                     **  Inclusive: A,B,C,D,D1,P,R0
2862                     **
2863                     ** Stk lvls:   3 (POP2DH)
2864                     **
2865                     ** History:
2866                     **
2867                     **    Date     Programmer             Modification
2868                     **  --------   ----------    ---------------------------------
2869                     **  03/01/83      NZ         Changed to always return non-
2870                     **                           negative value
2871                     **  02/28/83      NZ         Changed FLTRTN to do processing
2872                     **                           here
2873                     **  02/08/83      SC         Wrote routines
2874                     **
2875                     ********************************************************************
2876                     ********************************************************************
2877 F1E62 8822            NIBHEX 8822
```

```
2878 F1E66 7D80 =BINAND GOSUB  POP2DH        Pop 2 values
2879 F1E6A 0EF6         A=A&C  A
2880 F1E6E      FLTRTN
2881            *
2882            * Following instruction is not needed any more (overlooked on
2883            * 3/1/83 change)
2884            *
2885 F1E6E D3           D=0    A             If D[A]=0, then sign is positive
2886            *
2887 F1E70 AF2          C=0    W
2888 F1E73 D6           C=A    A
2889 F1E75 8E00         GOSUBL =FLOAT!       Convert to floating decimal
          00
2890 F1E7B 8D00 Fnrtn4  GOVLNG =FNRTN4
          000
2891            ********************************************************************
2892 F1E82 8822         NIBHEX 8822
2893 F1E86 7D60 =BINIOR GOSUB  POP2DH        Pop 2 numbers
2894 F1E8A 0EFE         A=A!C  A             Do an inclusive OR on them
2895 F1E8E 6FDF         GOTO   FLTRTN        Finish up
2896            ********************************************************************
2897 F1E92 8822         NIBHEX 8822
2898 F1E96 7D50 =BINEOR GOSUB  POP2DH        Pop 2 numbers
2899            *
2900            * A EOR C = (A and (not C)) or ((not A) and C)
2901            *
2902 F1E9A D8           B=A    A             Save A in B
2903 F1E9C FE           C=-C-1 A             C = not C
2904 F1E9E 0EF6         A=A&C  A             A = (A and (not C)
2905 F1EA2 DC           ABEX   A             B = (A and (not C), restore A
2906 F1EA4 FC           A=-A-1 A             A = not A
2907 F1EA6 DB           C=D    A             Restore C from D (POP2DH)
2908 F1EA8 0EF6         A=A&C  A             A = ((not A) and C)
2909 F1EAC 0EF8         A=A!B  A             A = A EOR C
2910 F1EB0 6DBF         GOTO   FLTRTN        Finish up
2911            ********************************************************************
2912 F1EB4 811          NIBHEX 811
2913 F1EB7 7CC8 =BINCMP GOSUB  Pop1n         Pop 1 number
2914 F1EBB 474          GOC    badtyp        (complex...error)
2915 F1EBE 7B60         GOSUB  fLTDH         Convert to HEX
2916 F1EC2 564          GONC   badinp        (range error)
2917 F1EC5 FC           A=-A-1 A             Do 1's complement
2918 F1EC7 66AF Fltrtn  GOTO   FLTRTN        Finish up
2919            ********************************************************************
2920 F1ECB 8822         NIBHEX 8822
2921 F1ECF 7420 =BIT    GOSUB  POP2DH
2922            *
2923            * C[A] is the value to check
2924            * A[A] is the bit position to check in value
2925            *
2926 F1ED3 D1           B=0    A
2927 F1ED5 E5           B=B+1  A             Use B[A] as the mask register
2928 F1ED7 CC   BIT10    A=A-1  A             Decrement bit count
2929 F1ED9 4D0          GOC    BIT20         Done making the mask
2930 F1EDC C5           B=B+B  A             Double the mask
```

```
2931 F1EDE 58F          GONC   BIT10          Go unless bit # too big
2932            *
2933            * If here, bit # was too big
2934            *
2935 F1EE1 20           P=     =eRANGE
2936 F1EE3 640F         GOTO   ErrorX
2937            *-
2938            *-
2939 F1EE7 0EF5 BIT20   C=C&B  A              Check if bit in that spot is set
2940 F1EEB D0           A=0    A
2941 F1EED 8AA          ?C=0   A
2942 F1EF0 7D           GOYES  Fltrtn         Return zero if C[A]=0
2943 F1EF2 E4           A=A+1  A
2944 F1EF4 52D          GONC   Fltrtn         Go always
2945            ****************************************************************
2946            ****************************************************************
2947            **
2948            ** Name:       POP2DH - Pop 2 numeric items, convert to HEX
2949            **
2950            ** Category:   LOCAL
2951            **
2952            ** Purpose:
2953            **      Pop two numbers off the stack and convert them to hex
2954            **
2955            ** Entry:
2956            **      P=0
2957            **      D1 points to the top of the stack
2958            **      Two numbers on the top of the stack
2959            **
2960            ** Exit:
2961            **      A[A] is the first number on the stack
2962            **      C[A] and D[A] are the second number on the stack
2963            **      Exits through ERRORX with eNNUMR if complex number,
2964            **        eRANGE if not in [0...2^20-1]
2965            **      Carry clear
2966            **
2967            ** Calls:      POP2N,FLTDH,<ERRORX>
2968            **
2969            ** Uses.......
2970            **  Inclusive: A,B,C,D,D1,P
2971            **
2972            ** Stk lvls:   2 (POP2N)
2973            **
2974            ** History:
2975            **
2976            **    Date      Programmer              Modification
2977            **  --------    ----------    ---------------------------------
2978            **  03/01/83      NZ          Added check for FLTDH error
2979            **  02/09/83      SC          Wrote routine
2980            **
2981            ****************************************************************
2982            ****************************************************************
2983 F1EF7 8F00 POP2DH  GOSBVL =POP2N         Pop 2 numbers
           000
2984 F1EFE 04           SETHEX
```

```
2985 F1F00 5D0          GONC   POP2D1        Go if no complex values
2986               *
2987 F1F03 20   badtyp  P=     =eNNUMR       Error...not numeric
2988 F1F05 62EE POP2ER  GOTO   ErrorX
2989               *-
2990               *-
2991 F1F09 20   badinp  P=     =eRANGE       Out of range error
2992 F1F0B 59F          GONC   POP2ER        Go always
2993               *-
2994               *-
2995               *
2996               * C[W] is the first number on stack
2997               * A[W] is the second number on stack
2998               *
2999 F1F0E AFF  POP2D1  CDEX   W              D=first number
3000 F1F11 7810         GOSUB  fLTDH         Convert second number to HEX
3001 F1F15 53F          GONC   badinp        Out of range or negative
3002 F1F18 AFE          ACEX   W
3003 F1F1B AFF          CDEX   W             D=second number, C=first number
3004 F1F1E AFE          ACEX   W             A=first number
3005 F1F21 7800         GOSUB  fLTDH         Convert first number to HEX
3006 F1F25 53E          GONC   badinp        Out of range or negative
3007 F1F28 AFB          C=D    W             C,D=second number,A=first number
3008 F1F2B 03           RTNCC
3009               *-
3010               *-
3011 F1F2D 8D00 =fLTDH  GOVLNG =FLTDH
          000
3012 F1F34              END
```

```
 #Timeo  Ext                    -  1413
 ASGN00  Abs  989678 #F19EE  -  1880  1864  1888
 ASGN04  Abs  989689 #F19F9  -  1885  1876
 ASGN05  Abs  989751 #F1A37  -  1920  1911
 ASGN10  Abs  989765 #F1A45  -  1925  1928
 ASGN15  Abs  989817 #F1A79  -  1962  1954
 ASGN20  Abs  989848 #F1A98  -  1981  2022
 ASGN25  Abs  989855 #F1A9F  -  1983  2017
 ASGN30  Abs  989859 #F1AA3  -  1986  1982
 ASGN40  Abs  989893 #F1AC5  -  1998  1987
 ASGN45  Abs  989921 #F1AE1  -  2004  2000
 ASGN50  Abs  989923 #F1AE3  -  2008  2002
 ASGNER  Abs  989970 #F1B12  -  2026  1990  1992  1994  2020
=ASGNI0  Abs  989645 #F19CD  -  1850
 ASGNd   Ext                    -  1848
 ASGNda  Abs  989992 #F1B28  -  2035  1880  2029
 ASGNeR  Abs  989972 #F1B14  -  2027  1955
 ASGNp   Ext                    -  1849
 ASLC2   Ext                    -  2009
 ASLC4   Ext                    -   914
 ASLC5   Ext                    -  1625
 ASRC2   Ext                    -  1998
 ASRC4   Ext                    -   826   907  1017
 ASRC5   Ext                    -  1630
 BAKCHR  Ext                    -  2003
 BF2DSP  Ext                    -  1612  1666
=BINAND  Abs  990822 #F1E66  -  2878
=BINCMP  Abs  990903 #F1EB7  -  2913
=BINEOR  Abs  990870 #F1E96  -  2898
=BINIOR  Abs  990854 #F1E86  -  2893
=BIT     Abs  990927 #F1ECF  -  2921
 BIT10   Abs  990935 #F1ED7  -  2928  2931
 BIT20   Abs  990951 #F1EE7  -  2939  2929
 BLANKC  Ext                    -  1592
 BSERR   Ext                    -  1912
 CHKASN  Ext                    -    78
 CHKMAS  Ext                    -   593  1035
 CKLOP#  Ext                    -  1106  1776
 CKmode  Ext                    -  1254
=CLEAR   Abs  988549 #F1585  -  1227
 CLEAR+  Abs  988616 #F15C8  -  1254  1245
 CLEAR.  Abs  988607 #F15BF  -  1248  1233
 CLEAR1  Abs  988659 #F15F3  -  1268  1260
 CLEAR2  Abs  988683 #F160B  -  1280  1272
 CLEAR3  Abs  988729 #F1639  -  1302  1300
 CLEAR4  Abs  988736 #F1640  -  1304  1294
 CLEARc  Abs  988556 #F158C  -  1228  1111  1148  1185
 CLEARd  Ext                    -  1225
 CLEARl  Abs  988578 #F15A2  -  1235  1107
 CLEARp  Ext                    -  1226
 CLEARs  Abs  988742 #F1646  -  1308  1266  1302
 CSLC12  Ext                    -  2281
 CSLC2   Ext                    -   847
 CSLC3   Ext                    -   694
 CSLC4   Ext                    -  1329
```

```
 CSRC4   Ext                 -  1333
 CloseR  Ext                 -   708
 Cslc4   Abs  988790 #F1676  -  1329   650   681   722
 Cslc5   Abs  988787 #F1673  -  1328   638   838   924  2477
 Csrc4   Abs  988797 #F167D  -  1333   614   653
 Csrc5   Abs  988794 #F167A  -  1332   636   909   922  2487
 D1=DSP  Ext                 -  1894
 D1=DST  Ext                 -   463  1728  1788
 D1=DSX  Ext                 -   460
=D1=SDO  Abs  988816 #F1690  -  1343  1102  1228  1291  1591
=D1=SRO  Abs  988807 #F1687  -  1339   277  1255  1268  1284
 DDL     Ext                 -  1325
 DDT     Ext                 -   713
=DEVID   Abs  990009 #F1B39  -  2078
 DEVID1  Abs  990033 #F1B51  -  2097  2087
 DEVID2  Abs  990043 #F1B5B  -  2105  2112
 DEVID3  Abs  990065 #F1B71  -  2115  2106
 DEVID4  Abs  990085 #F1B85  -  2128  2126
 DEVIDe  Abs  990102 #F1B96  -  2134  2079  2096
 DEVP10  Abs  990404 #F1CC4  -  2459  2427
 DEVP20  Abs  990472 #F1D08  -  2499  2441
 DEVP23  Abs  990485 #F1D15  -  2502
 DEVP25  Abs  990495 #F1D1F  -  2512  2499  2502
 DEVP30  Abs  990513 #F1D31  -  2524  2516
=DEVPAR  Abs  990341 #F1C85  -  2421  2078  2184  2251  2339
=DEVPR$  Abs  990411 #F1CCB  -  2464
 DEVPcc  Abs  990511 #F1D2F  -  2521  2504
 DEVTY4  Abs  990305 #F1C61  -  2353  2295  2359
 DEVTY5  Abs  990319 #F1C6F  -  2358  2194  2296  2345  2349
=DEVTYP  Abs  990268 #F1C3C  -  2339
 DEVTYe  Abs  990337 #F1C81  -  2368  2340  2347
 DEVTYx  Abs  990299 #F1C5B  -  2352  2210
 DISPI+  Abs  987468 #F114C  -   418   465
=DISPIS  Abs  987458 #F1142  -   408
 Ddl     Abs  988781 #F166D  -  1325   709   719
 Ddt     Abs  987901 #F12FD  -   713   706
 DdtXgT  Abs  987899 #F12FB  -   712
 ENDFN   Ext                 -  2129  2353
 ENDST   Ext                 -  1305
 ENDTAP  Ext                 -   890
 EOLLEN  Ext                 -   338
 ERRORX  Ext                 -  1277
 Endst   Abs  988736 #F1640  -  1305   420  1042  1806  1983
 ErrorX  Abs  990696 #F1DE8  -  2724  2706  2715  2717  2780  2936  2988
 Errorx  Abs  988677 #F1605  -  1277   237   359   519   873  1049  1238  1253
                             1265  1267  1303  1419  1559  1809  2032  2134
                             2299  2368  2579  2724
 F->SCR  Ext                 -   690
 FCNRT1  Abs  990591 #F1D7F  -  2606  2594  2829
=FIND    Abs  990171 #F1BDB  -  2251
 FIND10  Abs  990253 #F1C2D  -  2290  2287
 FINDER  Abs  990261 #F1C35  -  2299  2252
 FINDer  Abs  990175 #F1BDF  -  2252  2185  2198  2204
 FIXSPC  Ext                 -   468   512   739  2444  2528
 FLOAT!  Ext                 -  2352  2608  2889
```

```
 FLTDH   Ext                    -  3011
 FLTRTN  Abs  990830 #F1E6E     -  2880  2895  2910  2918
 FNDCH-  Ext                    -  1237
 FNDCHK  Ext                    -  2720
 FNRTN1  Ext                    -  2610
 FNRTN4  Ext                    -  2890
 FORMAT  Ext                    -  1040
 FUNCDO  Ext                    -   116   125   162
 FUNCRO  Ext                    -  1619  1665
 Fltrtn  Abs  990919 #F1EC7     -  2918  2942  2944
 Fndchk  Abs  990688 #F1DE0     -  2720  1483  2702
 Fnrtn-  Abs  990576 #F1D70     -  2600  2781
 Fnrtn.  Abs  990603 #F1D8B     -  2609  2601
 Fnrtn1  Abs  990607 #F1D8F     -  2610  2131
 Fnrtn4  Abs  990843 #F1E7B     -  2890  2355
 GADRRM  Ext                    -  2436
 GDIRST  Ext                    -   598
 GETDID  Ext                    -   331   429   506   734  1252
 GETDIR  Ext                    -   883
 GETDIX  Ext                    -  2483
 GETDR"  Ext                    -   604   802
 GETDR+  Ext                    -   670
 GETHEX  Ext                    -  1006
 GETHSS  Ext                    -  2824
 GETID+  Ext                    -  2091
 GETLPe  Abs  990694 #F1DE6     -  2723  2704
=GETLPs  Abs  990634 #F1DAA     -  2699  2583  2651  2774
 GETMBX  Ext                    -   215  1295
 GETPIL  Ext                    -   970
 GETSTR  Ext                    -  1854
=GETZER  Abs  988147 #F13F3     -   895   820   828
 GHEXB+  Ext                    -  2705
 GLOOP#  Ext                    -  1391
 GT2BYT  Ext                    -   903
 GTYPE   Ext                    -  2346
 Gt2byt  Abs  988162 #F1402     -   903   808   896   898
 HTOD    Ext                    -  1589  2260  2270
 I/OALL  Ext                    -  1910
 I/ODAL  Ext                    -  2037
=I/odal  Abs  989999 #F1B2F     -  2037
 IDIV    Ext                    -  1466
 INITLP  Abs  988282 #F147A     -   988   990
 INITX0  Abs  988296 #F1488     -   998   982
 INITX1  Abs  988392 #F14E8     -  1033   992
 INITX2  Abs  988423 #F1507     -  1048
 INITXE  Abs  988425 #F1509     -  1049  1007  1034  1036  1041
 INITXF  Abs  988332 #F14AC     -  1007   971  1011
=INITXQ  Abs  988246 #F1456     -   966
 INITd   Ext                    -   964
 INITp   Ext                    -   965
 IOp     Ext                    -  1563
 IS-DSP  Ext                    -   409   457
 IS-PRT  Ext                    -    73   425
 LCL10   Abs  988486 #F1546     -  1110  1101
 LEXPIL  Ext                    -  1096
```

```
 LIST10  Abs  989170 #F17F2 -  1579  1583
 LIST20  Abs  989302 #F1876 -  1618  1667
 LIST30  Abs  989437 #F18FD -  1656  1653
 LIST50  Abs  989469 #F191D -  1670  1647
=LISTIO  Abs  989139 #F17D3 -  1564
 LISTnb  Abs  989120 #F17C0 -  1557  1567
 LOAD-1  Abs  990326 #F1C76 -  2362  2358  2600
=LOCAL   Abs  988439 #F1517 -  1085
 LOCALd  Ext                -  1083
 LOCALp  Ext                -  1084
 LOOPST  Ext                -  1721  1782  1936
 LoopOK  Ext                -   222  1731  1791
 MLFFLG  Ext                -  2452
 MLFG=F  Abs  990382 #F1CAE -  2451  2500  2699
 MOVEFL  Ext                -   864
 MTYL    Ext                -  1316
 MeTalk  Abs       9 #00009 -    66   130   133   145
 Mtyl    Abs  988763 #F165B -  1316   151   716
 NXTCHR  Ext                -  1322
 NXTENT  Ext                -   629   652   916
 NXTSTM  Ext                -  1494
 Nxtchr  Abs  988775 #F1667 -  1322  1875  1981  1991  1999  2016
 Nxten+  Abs  988168 #F1408 -   906   869
 Nxten-  Abs  988182 #F1416 -   910   882
=OFFIO   Abs  989483 #F192B -  1708
 OFFIO1  Abs  989510 #F1946 -  1721  1711
 OFFIO2  Abs  989553 #F1971 -  1734  1718
 OFFIOd  Ext                -  1562  1706
 OFFIOp  Ext                -  1707
 ONINTR  Ext                -  1715
 OUTPTt  Ext                -   285   347
=OUTPUT  Abs  987372 #F10EC -   331
 OUTPd   Ext                -   329
 OUTPer  Abs  987444 #F1134 -   359   332
 OUTPp   Ext                -   330
 Offed   Ext                -  1724
=PACK    Abs  987974 #F1346 -   794
 PACK00  Abs  987978 #F134A -   795   736
 PACK10  Abs  987987 #F1353 -   802   870
 PACK20  Abs  987993 #F1359 -   803   884
 PACK30  Abs  988016 #F1370 -   816   812
 PACK40  Abs  988120 #F13D8 -   876   834
 PACK90  Abs  988134 #F13E6 -   890   509   813   926
=PACKD   Abs  987605 #F11D5 -   506
 PACKd   Ext                -   504   792
 PACKeR  Abs  988036 #F1384 -   829   795   803   821
 PACKer  Abs  988116 #F13D4 -   873   829   856   858   865   891   913
 PACKfx  Abs  987949 #F132D -   734   794
 PACKp   Ext                -   505   793
 PBF->C  Abs  987862 #F12D6 -   701   660
=PDIR    Abs  987629 #F11ED -   592   507   735
 PDIR10  Abs  987670 #F1216 -   604   662
 PDIR20  Abs  987679 #F121F -   606   672
 PDIR22  Abs  987701 #F1235 -   618   607
 PDIR24  Abs  987749 #F1265 -   640   631
```

```
 PDIR30  Abs  987803 #F129B -   668   643   655
 PDIR90  Abs  987821 #F12AD -   678   617   627
 PDIR92  Abs  987842 #F12C2 -   686   679
 PDIR95  Abs  987855 #F12CF -   693   685
 PDIRBF  Abs  987907 #F1303 -   716   647   691
 PILCNF  Ext                -   419  1805
 POP1N   Ext                -  1497
 POP2D1  Abs  990990 #F1F0E -  2999  2985
 POP2DH  Abs  990967 #F1EF7 -  2983  2878  2893  2898  2921
 POP2ER  Abs  990981 #F1F05 -  2988  2992
 POP2N   Ext                -  2983
=PRASCI  Abs  987263 #F107F -   211   170
 PRASER  Abs  987310 #F10AE -   232   224
 PRASEX  Abs  987297 #F10A1 -   225   233
=PREND   Abs  987319 #F10B7 -   272   210
 PREND1  Abs  987354 #F10DA -   289   286
 PRENDE  Abs  987360 #F10E0 -   294   283
=PREXT   Abs  987095 #F0FD7 -    99    94
 PRINT*  Ext                -   356
=PRNT00  Abs  987499 #F116B -   426   415
 PRNT45  Abs  987551 #F119F -   452   438
 PRNT50  Abs  987478 #F1156 -   420   459
 PRNTER  Abs  987625 #F11E9 -   519   443   445   508
=PRNTIS  Abs  987492 #F1164 -   425
 PRNTSd  Ext                -   406   423
 PRNTSp  Ext                -   407   424
=PRTIS   Abs  987041 #F0FA1 -    72
 PRTIS"  Abs  987176 #F1028 -   138   136
=PRTIS+  Abs  987054 #F0FAE -    74
 PRTIS,  Abs  987166 #F101E -   134   132
 PRTIS-  Abs  987097 #F0FD9 -   102    93
 PRTIS0  Abs  987114 #F0FEA -   116   139   152
 PRTIS1  Abs  987126 #F0FF6 -   119    84
 PRTIS2  Abs  987136 #F1000 -   125    79    89
 PRTIS4  Abs  987223 #F1057 -   162   154   161
 PRTIS5  Abs  987243 #F106B -   168   167   170
 PRTIS@  Abs  987201 #F1041 -   151   146
=PRTISc  Abs  987034 #F0F9A -    68  1270
 PRTISe  Abs  987060 #F0FB4 -    76    69
 PRTS00  Abs  987205 #F1045 -   152   148
 PRTS01  Abs  987208 #F1048 -   153   144
 PT2BYT  Ext                -   848
 PUTC    Ext                -  1310
 PUTDR"  Ext                -   728
 PUTDR#  Ext                -   855
 PUTE    Ext                -  1492
 PUTGF-  Ext                -  2587
 PhyEOD  Abs        0 #00000 -   611   612   618   678
 Popin   Abs  989063 #F1787 -  1497  1501  2432  2703  2913
 Putc    Abs  988751 #F164F -  1310  1487
 Putd    Ext                -   725
 R&CV10  Abs  990583 #F1D77 -  2604  2599
 R&CVER  Abs  990527 #F1D3F -  2579  2584  2588  2652
 R&CVEu  Abs  990520 #F1D38 -  2576  2590
 RD&CVT  Abs  990548 #F1D54 -  2587  2655
```

```
=READDC  Abs  990617 #F1D99 -  2651
=READIN  Abs  990534 #F1D46 -  2583
 READRG  Ext                -  2203
=REMOTE  Abs  988528 #F1570 -  1184
 REMOTd  Ext                -  1182
 REMOTp  Ext                -  1183
=REST10  Abs  989573 #F1985 -  1782  1882
 REST1A  Ext                -  1019
 REST2C  Ext                -  1015  1432  1460
 RESTD0  Ext                -   433
 RESTD1  Ext                -  1012
=RESTIO  Abs  989567 #F197F -  1776
 RESTd   Ext                -  1774
 RESTp   Ext                -  1775
 REVPOP  Ext                -  2463
 Rester  Abs  989631 #F19BF -  1809  1804
 SAVE1A  Ext                -  1005
 SAVE2C  Ext                -  1336
 SAVED0  Ext                -   427
 SAVED1  Ext                -  1001
 SAVEIT  Ext                -  1319
 SEEKA   Ext                -   703
 SPOL05  Abs  990121 #F1BA9 -  2194  2206
 SPOL10  Abs  990125 #F1BAD -  2197  2190
=SPOLL   Abs  990109 #F1B9D -  2184
 STAN10  Abs  988883 #F16D3 -  1407  1396
 STAN20  Abs  988907 #F16EB -  1422  1399
 STAN30  Abs  989002 #F174A -  1473  1471
 STAN40  Abs  989012 #F1754 -  1477  1404  1415  1444
=STANBY  Abs  988847 #F16AF -  1391
 STANDd  Ext                -  1389
 STANDp  Ext                -  1390
 STANeR  Abs  988961 #F1721 -  1454  1484  1488  1493
 STANer  Abs  988903 #F16E7 -  1419  1431  1454
 STANra  Abs  988901 #F16E5 -  1418  1476
 STANsb  Abs  989069 #F178D -  1500  1430  1453
 STANsr  Abs  989116 #F17BC -  1520  1512  1514  1516
 START   Ext                -   138  1033  2501
 START-  Ext                -  1803  1953
 STAT10  Abs  990719 #F1DFF -  2793  2778
 STAT21  Abs  990733 #F1E0D -  2799  2797
 STAT22  Abs  990741 #F1E15 -  2802  2800
 STAT23  Abs  990753 #F1E21 -  2807  2805
 STAT24  Abs  990761 #F1E29 -  2810  2808
 STAT25  Abs  990775 #F1E37 -  2816  2814
 STAT26  Abs  990783 #F1E3F -  2819  2817
 STAT27  Abs  990791 #F1E47 -  2822  2820
 STAT30  Abs  990814 #F1E5E -  2829  2826
=STATUS  Abs  990703 #F1DEF -  2774
 STMTD0  Ext                -  1257  1343  1611
 STMTR0  Ext                -   340  1339
 STMTR1  Ext                -   280   333
 SWAP01  Ext                -   434
 Save2c  Abs  988801 #F1681 -  1336   976  1003  1429  1440
 SaveIt  Abs       6 #00006 -    65   129   137   153
```

```
 Saveit  Abs  988769 #F1661 -  1319   288   336   452  1288
 SetBP   Ext                -   718
 TRESD0  Ext                -  1350
 TRESD1  Ext                -   110
=TRIGER  Abs  988507 #F155B -  1146
 TRIGd   Ext                -  1144
 TRIGp   Ext                -  1145
 TSAVD0  Ext                -  1347
 TSAVD1  Ext                -    72  1869  2469
 TSTAT   Ext                -  1313
 TSWAD1  Ext                -  1973  2008  2012
 Timout  Ext                -  1411
 Tresd0  Abs  988831 #F169F -  1350   225   337   977  1972  2130  2354  2609
 Tresd1  Abs  987108 #F0FE4 -   110   119   165  2493
 Tsavd0  Abs  988825 #F1699 -  1347   213  1868  2431  2700
 Tstat   Abs  988757 #F1655 -  1313   710   857
 UCRANG  Ext                -  1986  1993
 ULYL    Ext                -   147   707
 UNLPUT  Ext                -  1264
 UTLEND  Ext                -   293
 WRITIT  Ext                -   223
 WRTASC  Ext                -  1596  1637
 Write1  Ext                -   727
 XchgT   Ext                -   705   712
 YTML    Ext                -  2197
 bPILAI  Ext                -  1565  1909  2036
=bSERR   Abs  989744 #F1A30 -  1912
 badinp  Abs  990985 #F1F09 -  2991  2916  3001  3006
 badtyp  Abs  990979 #F1F03 -  2987  2914
 cATCH+  Ext                -  2001
 eABORT  Ext                -  1276
 eDSPEC  Ext                -   444  2026  2508
 eDTYPE  Ext                -  1048
 eNMBOX  Ext                -  2779
 eNNUMR  Ext                -  2723  2987
 eNOASN  Ext                -  1557
 eNOFND  Ext                -  2515
 ePARSE  Ext                -  1558
 ePIL    Ext                -  2512  2578
 eRANGE  Ext                -   899   912  1009  1418  1520  2713  2935  2991
 eUNEXP  Ext                -  2577
 eXPEXC  Ext                -  1500
=fLTDH   Abs  991021 #F1F2D -  3011  1513  2915  3000  3005
 hsLPRQ  Ext                -  2825
 i/OFND  Ext                -  1566
 mCMDf   Ext                -  1308
 mREADC  Ext                -  2653
 mREADI  Ext                -  2585
 mSETIC  Ext                -  1485
 mSST    Ext                -  2202
 mSTO@5  Ext                -  1490
=nXTSTM  Abs  989056 #F1780 -  1494   892  1674  1734
 pDIAGL  Ext                -  2589
 sCA     Abs       5 #00005 -  2786  2821
 sEar    Abs       2 #00002 -  2789  2809
```

```
 sLA      Abs       6 #00006 -  2785  2818
 sLLout   Abs       0 #00000 -  2791  2798
 sReadd   Ext                 -  1797  1952
 sRemot   Abs       1 #00001 -  2790  2801
 sSRQR    Abs       3 #00003 -  2788  2827
 sSTK     Ext                 -  1863  2423
 sStanb   Abs       7 #00007 -  2784  2806
 sTA      Abs       4 #00004 -  2787  2815
 tCOMMA   Ext                 -   979  1231  1442
 tLOCKO   Ext                 -  1097
 tOFF     Ext                 -  1394
 tON      Ext                 -  1397
 tXWORD   Ext                 -  1095  1709
```

Input Parameters

  Source file name is NZ&BAS::MS

  Listing file name is NZ/BAS:TI:ML::-1

  Object file name is NZ%BAS:TI:MS::-1

```
                                    111111
                          0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *        SSS    CCC    &     EEEEE  N   N  TTTTT
 2            *       S   S  C   C  & &    E      N   N    T
 3            *       S      C      & &    E      NN  N    T
 4            *        SSS   C       &     EEEE   N N N    T
 5            *           S  C      & & &  E      N  NN    T
 6            *       S   S  C   C  &  &   E      N   N    T
 7            *        SSS    CCC    && &  EEEEE  N   N    T
 8            *
 9                    TITLE  ENTER Execution <840113.1057>
10 F1F34             ABS    #F1F34        TI%HP6 address (fixed)
11            *
12            Array   EQU    1
13            String  EQU    2
14            Cmplex  EQU    3
15            Endfrm  EQU    3
16            MltItm  EQU    4
17            Memerr  EQU    4
18            BytCnt  EQU    5
19            KorH    EQU    5
20            Sign    EQU    6
21            Trash   EQU    6
22            ChrTrp  EQU    7
23            ****************************************************************
24            ****************************************************************
25            **
26            ** Name:       hENTER - Poll handler for the pENTER poll
27            **
28            ** Category:   POLL
29            **
30            ** Type:       POLL
31            **
32            ** Purpose:
33            **      To read data from HP-IL and put it on Math stack
34            **
35            ** Entry:
36            **      B[A] = Poll number.
37            **      HEX mode.
38            **      P=0.
39            **      MTHSTK=FORSTK  (Math stack is collapsed to FORSTK)
40            **
41            **      R1[A]=HP-IL address (device's location relative to the
42            **              controller)
43            **
44            **      S5 (BytCnt):
45            **           1:Read a specified number of characters
46            **               A[A] is the number of characters to read
47            **           0:Terminate by END frame or terminating char match
48            **               A[B] is the terminating character
49            **
50            **      S6 (Trash):
51            **           1:Ignore the data which is read
52            **           0:Save the data which is read on the stack
53            **
54            **      S7 (ChrTrp):
55            **           1:Detect a special character in incoming data
```

```
                **              R2[B] is the character to be detected
                **              If R2[3:2]=00, ignore the character;
                **              otherwise replace the character with R2[3:2]
                **          0:No special character processing
                **
                **      If system flag -23 is set:
                **        Terminate by ETO, terminating character is ignored
                **
                **        If S5 (BytCnt)=0, S6 (Trash)=0, and S-R0-3[0]>2 (the
                **          destination is a string), then S-R1-1[3:0] and R3[A]
                **          are the maximum number of chars to read before
                **          interrupting the conversation with an NRD.
                **          R3[S] must not be "F". (R3[4]=0)
                **
                **        If S5 (BytCnt)=1 or S6 (Trash)=1, then flag -23 has
                **          no effect other than to terminate on an ETO instead
                **          of the terminator character.
                **
                **        If { S-R0-3[0]<=2 (not string dest) and S5 (BytCnt)=0 }
                **          or { in device mode (not controller) },
                **          then flag -23 has no effect (it is ignored).
                **
                **
                ** Exit:
                **      HEX mode.
                **      XM=0.
                **      Carry clear:
                **        AVMEME points to the last character read
                **        FORSTK points to first char read + 2
                **        Number of chars read = ((FORSTK) - (AVMEME))/2
                **        S4 (Memerr)=0
                **      Carry set:
                **        S4 (Memerr)=1: Insufficient memory (Need to load eMEM)
                **        S4 (Memerr)=0: C[3:0] is the error code
                **
                ** Calls:      D1=AVE,RDST01,<ERROR>,<AVE=D1>
                **
                ** Uses:
                **  Inclusive: A-D,D0,D1,P,R1,R2,ST[5:0]
                **
                ** Stk Lvls:   5 (RDST01)
                **
                ** History:
                **
                **    Date     Programmer          Modification
                **  --------   ----------   --------------------------------
                **  12/13/83      NZ        Updated documentation
                **  07/26/83      SC        Wrote routine
                **
                ****************************************************************
                ****************************************************************
F1F34 11A       =hENTER C=R2                Get special char (for ChrTrp=1)
F1F37 D5                B=C    A            Place in B[B], B[3:2]
F1F39 7317              GOSUB  D1=mstk      Set D1 to the top of the math stack
F1F3D 70C3              GOSUB  RDST01       Read the characters...
```

```
111 F1F41 580          GONC   pENTR1        ...No error (leave AVMEME at stack)
112 F1F44 8C00         GOLONG =ERROR        Error (set up C[3:0])
          00
113             *-
114             *-
115 F1F4A 6072 pENTR1  GOTO   aVE=D1        Carry clear, AVMEME updated
116             ****************************************************************
117             ****************************************************************
118             **
119             ** Name:       ENTER - Execute the ENTER statement
120             **
121             ** Category:   STEXEC
122             **
123             ** Purpose:
124             **      Execute the ENTER statement to read data from the loop
125             **
126             ** Entry:
127             **      D0 points to the device specifier
128             **      P=0
129             **
130             ** Exit:
131             **      Through either NXTSTM or BSERR
132             **
133             ** Calls:      GETDID,DEVADR,SAVEIT,TRESDO,CHKEOL,NXTDST,RED-LF,
134             **             STRPcr,CS=TYP,STRHED,REV$,D1MSTK,GETNUM,STOSUB,
135             **             FSTK-7,AVE=D1,RESTDO,NXTDS+,<NXTSTM>,<USING>,
136             **             <ERRORX>,<getEOL>
137             **
138             ** Uses.......
139             **  Inclusive: A,B,C,D,R0-R4,D0,D1,P,STMTxx,ST[11:0],FUNCxx,
140             **             All RAM EXPEXC is permitted to use
141             **
142             ** Stk lvls:   7 (GETDID)(STOSUB)
143             **
144             ** History:
145             **
146             **    Date     Programmer          Modification
147             **  --------   ----------   --------------------------------
148             **  12/20/83      NZ        Packed 3 places to get room for
149             **                          bug fix in GETNUM (locations are
150             **                          marked with a "+" in col. 29)
151             **  12/15/83      NZ        Added documentation
152             **  04/01/82      SC        Wrote routine
153             **
154             ****************************************************************
155             ****************************************************************
156 F1F4E 0000            REL(5) =OUTPd
          0
157 F1F53 0000            REL(5) =ENTERp
          0
158 F1F58 8E00 =ENTER  GOSUBL =GETDID       Get Device specifier
          00
159 F1F5E 431          GOC    ENTREX        Error...P,C[0] are error code
160             *
161             * D0 points to the mailbox, FUNCD0 contains the PC value
```

```
162                *
163 F1F61 96F      ENTRY   ?D#0    B           Is the address non-zero?
164 F1F64 D2               GOYES   GETD10      Yes...valid address
165 F1F66 2F               P=      15          No...check for LOOP (not NULL)
166 F1F68 300              LC(1)   =DsLoop
167 F1F6B 943              ?C=D    S           Is this "LOOP"?
168 F1F6E 02               GOYES   GETD09      Yes...accept it
169 F1F70 20               P=      =eDSPEC     No...must be "NULL"
170 F1F72 8C00 ENTREX      GOLONG  =ERRORX     Error exit for P, C[0]=error code
          00
171                *-
172                *-
173 F1F78 49F      RTNCHK  GOC     ENTREX      If carry, detected an error
174                *
175                * Delete the buffer (if any) created by SAVEIT before finishing
176                *
177 F1F7B 7342 ENTdel  GOSUB  DEVADR       Set D1 to the device specifier
178 F1F7F AF2          C=0     W           Replace it with zero (no device)
179 F1F82 8E00         GOSUBL  =SAVEIT     SAVEIT deletes any old buffer
          00
180 F1F88 8C00 ENTRTN  GOLONG  =nXTSTM     Finished!
          00
181                *-
182                *-
183 F1F8E AC2  GETD09  C=0     S           This is LOOP...don't make a buffer
184 F1F91 7D22 GETD10  GOSUB   DEVADR      Set (MTHSTK) = (FORSTK) - 7
185 F1F95 8E00         GOSUBL  =SAVEIT     Save device specifier on MTHSTK
          00
186 F1F9B 8E00         GOSUBL  =TRESDO     Restore PC (saved by GETDID)
          00
187 F1FA1 161          D0=D0+  2           Skip the t@ used to terminate spec
188 F1FA4 14A          A=DAT0  B
189 F1FA7 3100         LC(2)   =tUSING
190 F1FAB 966          ?A#C    B           Is this ENTER ... USING?
191 F1FAE 51           GOYES   ENT120      No...continue with ENTER
192 F1FB0 1F00         D1=(5)  =MLFFLG     Yes...zero MLFFLG, device to prevent
          000
193 F1FB7 D2           C=0     A           .CKINFO from doing anything bad when
194 F1FB9 14D          DAT1=C  B           .USING calls it
195 F1FBC 8D00         GOVLNG  =USING
          000
196                *-
197                *-
198 F1FC3 8F00 ENT120  GOSBVL  =CHKEOL      Are there any variables specified?
          000
199 F1FCA 460          GOC     ENT130      Yes...read and store
200                *
201                * ENTER statement has no destination variable:
202                * just skip to end of line and return.
203                *
204 F1FCD 6517         GOTO    getEOL
205                *-
206                *-
207 F1FD1 7C52 ENT130  GOSUB   NXTDST       Set up next destination and loop
208 F1FD5 55A          GONC    ENTdel      Reached end of line...done
```

```
209 F1FD8 7803 ENT150  GOSUB  RED-LF        Read until <Lf>
210 F1FDC 580          GONC   ENT155        Good read...continue
211 F1FDF 8C8F         GOLONG REDCer        Error during read...exit with error
          80
212             *-
213             *-
214 F1FE5 845  ENT155  ST=0   KorH          This is not USING format "K" or "H"
215 F1FE8 844  ENT160  ST=0   MltItm        Not multiple items per data line
216 F1FEB 94C          ?A#0   S             Is flag -23 set?
217 F1FEE B0           GOYES  ENT180        Yes...keep all characters
218             *
219 F1FF0 873          ?ST=1  Endfrm        Was the last byte an END frame?
220 F1FF3 60           GOYES  ENT180        If so, don't strip off <CR>
221             *
222 F1FF5 7F36         GOSUB  STRPcr        Strip off trailing <cr> if present
223             *
224 F1FF9 78F1 ENT180  GOSUB  CS=TYP        Returns carry set if numeric type
225 F1FFD 4A4          GOC    ENT220        Numeric variable...process it
226             *
227             * Destination is a string variable: make sure not to exceed the
228             * maximum string length.
229             *
230 F2000 864          ?ST=0  MltItm        Has another item been processed?
231 F2003 C1           GOYES  ENT190        No...continue
232             *
233             * A numeric item has been processed already (strings use up the
234             * entire line which has been read).  Processing a numeric item
235             * reverses the string on the stack, so we have to reverse it
236             * again to get back to original order.
237             *
238 F2005 7046         GOSUB  strhed        Put a header on to reverse the data
239 F2009 8E00         GOSUBL =rEV$         Reverse the string
          00
240 F200F 17F          D1=D1+ 16            Skip the header (16 nibbles)
241 F2012 137          CD1EX                Save D1 in C[A]
242 F2015 79A1         GOSUB  DEVADR        Set AVMEME back to FORSTK - 7
243 F2019 135          D1=C                 Restore D1
244 F201C 171          D1=D1+ 2             Skip the <Cr> that GETNUM added
245             *
246             * D1 points to the end of the string (lowest address)
247             *
248 F201F 133  ENT190  AD1EX                Save D1 in A[A]
249 F2022 7A26         GOSUB  D1mstk        Set D1 to AVMEME (=MTHSTK)
250 F2026 D6           C=A    A             Copy old D1 value to C[A]
251 F2028 133          AD1EX                Restore D1, set A[A] to AVMEME
252 F202B EE           C=A-C  A             C[A] is number of nibbles on stack
253 F202D 7164         GOSUB  A=SLEN        Recall maximum string length
254 F2031 C4           A=A+A  A             A[A] is the max length in nibbles
255 F2033 E2           C=C-A  A             Check if the data will fit in string
256 F2035 4A0          GOC    ENT200        Yes...do the assignment
257 F2038 133          AD1EX                No...throw away the excess chars
258 F203B CA           A=A+C  A             (C[A] is the number of extra nibs)
259 F203D 133          AD1EX
260 F2040 7506 ENT200  GOSUB  strhed        Put a string header on the data
261 F2044 6410         GOTO   ENT300        Go do the string assignment
```

```
262                *-
263                *-
264                *
265                * Destination is a numeric variable: try to get a number out of
266                * the data
267                *
268 F2048 7660 ENT220  GOSUB  GETNUM        Get a number, if possible
269 F204C 4B8          GOC    ENT150        No number, MltItm; read another line
270 F204F AF4  ENT250  A=B    W
271 F2052 1CF          D1=D1- 16
272 F2055 1517         DAT1=A W             Push number value onto the stack
273 F2059 865  ENT300  ?ST=0  KorH          Is this ENTER ... USING "K" or "H"?
274 F205C 60           GOYES  ENT302        No...store and loop back
275 F205E 6B01         GOTO   STOSUB        Yes...store and return to caller
276                *-
277                *-
278 F2062 7401 ENT302  GOSUB  STOSUB        Store the number
279 F2066 76E5         GOSUB  D1mstk        Set D1 to (MTHSTK)
280 F206A 7271         GOSUB  FSTK-7        Set D0 to (FORSTK) - 7
281 F206E 136          CD0EX                C[A] is (FORSTK) - 7
282 F2071 133          AD1EX                A[A] is (MTHSTK)
283 F2074 8E00         GOSUBL =RESTD0       Restore D0 from STMTD0
          00
284 F207A 8BE          ?A>=C  A             Any data left in line?
285 F207D 51           GOYES  ENT305        No...get next dest, read a line
286                *
287                * If there is exactly one character left on the stack, it must
288                * be the <Cr> GETNUM added to the string.
289                *
290 F207F CE           C=C-1  A
291 F2081 CE           C=C-1  A             Back up 2 nibbles
292 F2083 8B2          ?A<C   A             Any data left?
293 F2086 01           GOYES  ENT310        Yes...set up next dest, GOTO ENT180
294 F2088 131          D1=A                 No...
295 F208B 171          D1=D1+ 2             ...set D1 to bottom of stack
296 F208E 7921         GOSUB  aVE=D1        Set AVMEME to bottom of stack
297 F2092 6E3F ENT305  GOTO   ENT130        Get next destination, read line
298                *-
299                *-
300 F2096 7791 ENT310  GOSUB  NXTDST        Get next destination variable
301 F209A 460          GOC    ENT320        Got another destination...continue
302 F209D 6DDE         GOTO   ENTdel        No more variables...exit
303                *-
304                *-
305 F20A1 7D11 ENT320  GOSUB  DEVADR        Set AVMEME to (FORSTK) - 7
306 F20A5 135          D1=C                 Set D1 @ top of stack (from NXTDST)
307 F20A8 854          ST=1   MltItm        Set Multi-Item flag
308 F20AB 845          ST=0   KorH          Not ENTER ... USING "K" or "H"
309 F20AE 6A4F         GOTO   ENT180        Continue processing line
```

```
                        STITLE Convert string into a number
                *****************************************************************
                *****************************************************************
                **
                ** Name:        GETNUM - Convert data on stack into a number
                **
                ** Category:    LOCAL
                **
                ** Purpose:
                **      Skip over any non-digit chars and convert the ASCII
                **      digits into a floating number
                **
                ** Entry:
                **      P=0
                **      HEXMODE
                **      D1 points to the lowest-addressed character of the data
                **      ST[MltItm]=1:
                **        D1 points to first character of the string
                **      ST[MltItm]=0:
                **        D1 points to last character of the string
                **
                ** Exit:
                **      Carry clear:
                **        B[W] is the floating number value
                **      Carry set:
                **        No digit found and ST[MltItm]=1
                **
                ** Calls:       STRHED,REV$,AVE=D1,RANGEN,NUMSCN,TSAVD1,BLDCON,
                **              NRMCON,TRESD1
                **
                ** Uses.......
                **  Inclusive: A,B,C,D,R0,R2,D0,D1,P,FUNCD1,ST[6,3,2,1]
                **
                ** Stk lvls:   2 (NUMSCN)(STRHED)(REV$)(TSAVD1)(TRESD1)
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------    ------------------------------
                **  12/20/83        NZ        Packed, installed bug fix for
                **                            SR #0039-01070(2).  This is the
                **                            bug where ENTER of an underflow
                **                            or an overflow will destroy some
                **                            user flags and traps.  This bug
                **                            exists in version HPIL:1A.
                **  12/15/83        NZ        Updated documentation
                **  03/02/83        SC        Wrote routine
                **
                *****************************************************************
                *****************************************************************
F20B2 31D0  GETNUM  LCHEX  0D               Add a <Cr> as the last digit...
F20B6 1C1           D1=D1- 2                (if MltItm is set, it will be the
F20B9 14D           DAT1=C B                first digit, but will be skipped)
                *
F20BC 7985          GOSUB  strhed           Put a string header on data
```

```
365                *
366                * If not the first number of the input string, don't reverse
367                * the string - it already has been reversed the first time thru
368                *
369 F20C0 874              ?ST=1  MltItm        Is this the first time through?
370 F20C3 80               GOYES  GETN10        No...leave it alone (already done)
371 F20C5 8E00             GOSUBL =rEV$         Yes...reverse the string
           00
372 F20CB 171  GETN10      D1=D1+ 2             Skip the first byte of header
373 F20CE AF2              C=0    W
374 F20D1 147              C=DAT1 A             Read string length in nibbles
375 F20D4 81E              CSRB                 C[A] is string length in bytes
376 F20D7 D5               B=C    A             B[A] is number of bytes on stack
377 F20D9 17D              D1=D1+ 14            Position to first character
378 F20DC 846              ST=0   Sign          Initialize the sign
379 F20DF CD   GETN20      B=B-1  A             Check if string exhausted yet
380 F20E1 521              GONC   GETN40        No...check the character
381                *
382                * No digits found in the string.
383                * If ST[MltItm]=0, just return zero.
384                * If ST[MltItm]=1, pop the stack and return with carry set
385                *
386 F20E4 864              ?ST=0  MltItm        First number in string?
387 F20E7 80               GOYES  GETN30        Yes...return zero
388 F20E9 7EC0             GOSUB  aVE=D1        No...pop stack, set carry to
389 F20ED 02               RTNSC                indicate need to read more data
390                *-
391                *-
392 F20EF AF1  GETN30      B=0    W             Set up a  floating number zero
393 F20F2 03               RTNCC                Return, all OK
394                *-
395                *-
396 F20F4 14B  GETN40      A=DAT1 B             Read the next character
397 F20F7 8E00             GOSUBL =RANGEN       Is it in [0,9]?
           00
398 F20FD 502              GONC   GETN60        Yes...continue
399 F2100 31E2             LCASC  \.\           No
400 F2104 962              ?A=C   B             Is is a decimal point?
401 F2107 71               GOYES  GETN60        Yes...consider it a digit
402 F2109 856              ST=1   Sign          No...set sign initially negative
403 F210C 31D2             LCASC  \-\
404 F2110 962              ?A=C   B             Is it a minus sign?
405 F2113 50               GOYES  GETN50        Yes...leave sign negative
406 F2115 846              ST=0   Sign          No...set sign back to positive
407 F2118 171  GETN50      D1=D1+ 2             Position to next character
408 F211B 53C              GONC   GETN20        Go always
409                *-
410                *-
411 F211E AF1  GETN60      B=0    W             Initialize the number
412 F2121 841              ST=0   1             Clear these two statuses for
413 F2124 842              ST=0   2               NUMSCN (if not zero, then error)
414 F2127 118              C=R0                 Save R0 value...
415 F212A 10A              R2=C                 ...in R2
416 F212D 8F00             GOSBVL =NUMSCN       Scan the string for a number
           000
```

```
417 F2134 04          SETHEX              (NUMSCN leaves DEC mode)
418 F2136 8E00        GOSUBL =TSAVD1      Save D1 to save from BLDCON/NRMCON
          00
419 F213C 8F00        GOSBVL =BLDCON      Convert NUMSCN output to tokenized
          000
420 F2143 8F00        GOSBVL =NRMCON      Convert tokenized to floating num
          000
421 F214A 8E00        GOSUBL =TRESD1      Restore D1 from FUNCD1
          00
422 F2150 11A         C=R2                Restore R0 from R2
423 F2153 108         R0=C
424 F2156 AF8         B=A    W            [S] is garbage here
425 F2159 AC1         B=0    S            Set the sign positive initially
426 F215C 866         ?ST=0  Sign         Is the sign positive?
427 F215F 90          GOYES  GETN80       Yes...done
428 F2161 05          SETDEC              No...
429 F2163 A4D         B=B-1  S            Set sign negative
430 F2166 04          SETHEX
431 F2168 03  GETN80  RTNCC               Return, got a good string
```

```
                        STITLE Store item into variable
                *****************************************************************
                *****************************************************************
                **
                ** Name:       STOSUB - Subroutine to store into a variable
                **
                ** Category:   LOCAL
                **
                ** Purpose:
                **      Assign the value on the stack to the variable location
                **      indicated by Statement scratch RAM
                **
                ** Entry:
                **      P=0
                **      STMTR0 and STMTR1 set up as by DEST
                **      D1 points to top of stack
                **      R0[A] is the saved D1 value
                **
                ** Exit:
                **      P=0
                **      The item has been popped off the stack
                **      AVMEME is updated to new top of stack
                **      D1 restored from R0[A]
                **
                ** Calls:      AVE=D1,CSLC5,CSRC5,STORE,D1MSTK,POPMTH,<ENTST3>
                **
                ** Uses.......
                **  Inclusive: A,B,C,D,R0[15:5],R1,R2,R3[15:5],R4,D0,D1,P,
                **             RESREG,ST[11:8,5,3,0]
                **
                ** Stk lvls:   6 (STORE)
                **
                ** History:
                **
                **    Date     Programmer          Modification
                **  --------   ----------   -------------------------------
                **  12/02/83      NZ        Added documentation
                **  04/01/82      SC        Wrote routine
                **
                *****************************************************************
                *****************************************************************
F216A 7D40      STOSUB  GOSUB  aVE=D1        Set stack pointer to D1 value
                *
                * Need to save R0[A] and R3[A] from STORE...use R4[14:10] for
                * R3[A], R4[9:5] for R0[A]
                *
F216E 110               A=R0
F2171 11B               C=R3
F2174 8E00              GOSUBL =CSLC5        R3[A] now in C[9:5]
      00
F217A D6                C=A    A
F217C 8E00              GOSUBL =CSLC5        R0[A] in C[9:5], R3[A] in C[14:10]
      00
F2182 10C               R4=C                 Put it all in R4
F2185 1537              A=DAT1 W             Recall the value from the stack
```

```
485 F2189 8F00          GOSBVL =STORE        Store it
          000
486             *
487             * Now restore R0[A] and R3[A] from R4
488             *
489 F2190 11C           C=R4
490 F2193 8E00          GOSUBL =CSRC5
          00
491 F2199 108           R0=C
492 F219C 8E00          GOSUBL =CSRC5
          00
493 F21A2 10B           R3=C
494             *
495             * R0 and R3 are now restored...pop the item off the stack
496             *
497 F21A5 77A4 popstk   GOSUB  D1=stk        First set D1 to top of stack
498 F21A9 8F00          GOSBVL =POPMTH       Pop the item
          000
499 F21B0 7700          GOSUB  aVE=D1        Set AVMEME to new top of stack
500 F21B4 AF4           A=B    W             Copy B to A for popstk entry
501 F21B7 6717          GOTO   ENTST3        Finish it up
502             *-
503             *-
504 F21BB 8D00 =aVE=D1 GOVLNG =AVE=D1
          000
```

```
                        STITLE Utility routines
                *****************************************************************
                *****************************************************************
                **
                ** Name:       DEVADR - Collapse MTHSTK, D1 to FORSTK - 7
                **
                ** Category:   LOCAL
                **
                ** Purpose:
                **      Collapse MTHSTK to FORSTK - 7, leave D1 at (MTHSTK)
                **
                ** Entry:
                **      None
                **
                ** Exit:
                **      Carry clear
                **      MTHSTK at (FORSTK) - 7
                **      D1 at (MTHSTK)
                **
                ** Calls:      None
                **
                ** Uses.......
                **  Inclusive: A[A],D1
                **
                ** Stk lvls:   0
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------    ------------------------------
                **  12/15/83      NZ          Added documentation
                **  04/01/82      SC          Wrote routine
                **
                *****************************************************************
                *****************************************************************
F21C2 1F000 DEVADR  D1=(5) =FORSTK
              000
F21C9 143           A=DAT1 A              A[A] is FORSTK pointer
F21CC 1C4           D1=D1- 5              D1 points to MTHSTK
                *
                * SET (MTHSTK) = (FORSTK) - 7
                *
F21CF 133           AD1EX                 D1 is now (FORSTK)
F21D2 1C6           D1=D1- 7              D1 is (FORSTK) - 7
F21D5 133           AD1EX                 A[A] is (FORSTK)-7, D1 is MTHSTK
F21D8 141           DAT1=A A              Write out (FORSTK)-7 to MTHSTK
F21DB 133           AD1EX                 D1 is (FORSTK)-7
F21DE 03            RTNCC
                *****************************************************************
                *****************************************************************
                **
                ** Name:       FSTK-7 - Set D0 to (FORSTK) - 7 and read 5 nibs
                **
                ** Category:   LOCAL
                **
```

```
559             ** Purpose:
560             **      Set D0 to (FORSTK) - 7
561             **
562             ** Entry:
563             **      None
564             **
565             ** Exit:
566             **      D0 points to (FORSTK) - 7
567             **      C[A] is the data at D0
568             **      Carry clear
569             **
570             ** Calls:       None
571             **
572             ** Uses.......
573             **  Inclusive: C[A],D0
574             **
575             ** Stk lvls:   0
576             **
577             ** History:
578             **
579             **    Date      Programmer              Modification
580             **  --------    ----------    ---------------------------------
581             **  12/15/83       NZ         Added documentation
582             **  04/01/82       SC         Wrote routine
583             **
584             ********************************************************************
585             ********************************************************************
586 F21E0 1B00 FSTK-7  D0=(5) =FORSTK
          000
587 F21E7 146          C=DAT0 A
588 F21EA 134          D0=C
589 F21ED 186          D0=D0- 7              D0 is at (FORSTK)-7
590 F21F0 146          C=DAT0 A              C[A] is (D0)
591 F21F3 01           RTN                   Carry is clear from D0=D0-7 above
592             ********************************************************************
593             ********************************************************************
594             **
595             ** Name:        CS=TYP - Check if the destination is numeric
596             **
597             ** Category:    LOCAL
598             **
599             ** Purpose:
600             **      Check if the destination variable is of type numeric
601             **      or not
602             **
603             ** Entry:
604             **      S-R0-3 contains the variable type
605             **
606             ** Exit:
607             **      Carry set if numeric, else clear
608             **
609             ** Calls:       None
610             **
611             ** Uses.......
612             **  Inclusive: C[S],C[A]
```

```
                **
                ** Stk lvls:   0
                **
                ** History:
                **
                **    Date     Programmer          Modification
                **  --------   ----------   --------------------------------
                **  12/15/83      NZ        Added documentation
                **  04/01/82      SC        Wrote routine
                **
                ********************************************************************
                ********************************************************************
F21F5 136       CS=TYP  CDOEX               Save DO in C[A]
F21F8 1B00              DO=(5) =S-RO-3
      000
F21FF 1564              C=DATO S
F2203 136               CDOEX               Restore DO from C[A]
F2206 A4E               C=C-1  S
F2209 400               RTNC                C[S] was 0
F220C A4E               C=C-1  S
F220F 400               RTNC                C[S] was 1
F2212 A4E               C=C-1  S
F2215 01                RTN                 C[S] was 2 if carry set, else >2
                ********************************************************************
                ********************************************************************
                **
                ** Name:       AS=FTY - Read and clear image type flag (CHN#SV)
                **
                ** Category:   LOCAL
                **
                ** Purpose:
                **      Read contents of CHN#SV into A[S] and clear CHN#SV
                **
                ** Entry:
                **      None
                **
                ** Exit:
                **      Carry unchanged from entry
                **      A[S] is the old contents of CHN#SV
                **
                ** Calls:      None
                **
                ** Uses.......
                **  Inclusive: A[S],C[S]
                **
                ** Stk lvls:   0
                **
                ** History:
                **
                **    Date     Programmer          Modification
                **  --------   ----------   --------------------------------
                **  12/15/83      NZ        Added documentation
                **  04/01/82      SC        Wrote routine
                **
                ********************************************************************
```

```
667            ****************************************************************
668 F2217 1B00 AS=FTY  DO=(5) =CHN#SV
          000
669 F221E 1524         A=DATO S            Read the old value into A[S]
670 F2222 AC2          C=0    S
671 F2225 1544         DATO=C S            Clear CHN#SV (write a zero)
672 F2229 01           RTN                 Return, carry unchanged
```

```
673                     STITLE Get next dest. variable
674                 ****************************************************************
675                 ****************************************************************
676                 **
677                 ** Name:       NXTDST - Get the next destination variable
678                 **
679                 ** Purpose:
680                 **      Get next variable from variable list.
681                 **      The variable will be created if not yet exist.
682                 **
683                 ** Entry:
684                 **      D0 is the PC
685                 **      P=0
686                 **
687                 ** Exit:
688                 **      D0 is the PC
689                 **      Carry clear:
690                 **        Reached end of variable list
691                 **      Carry set:
692                 **        Variable on top of stack
693                 **        C,D1 point to top of stack (variable has been popped)
694                 **        AVMEME=D1
695                 **        S2=1 if string variable
696                 **          (S-R1-1[3:0]=Maximum string length)
697                 **
698                 **      Error exit if the variable is an array or complex number
699                 **      Error exit if insufficient memory to create new variable
700                 **      Error exit if encounter any error on the loop
701                 **
702                 ** Calls:      RESTDO,CHKEOL,MFLG=0,EXPEXC,NXTVA-,DIMST+,STKVCT,
703                 **             D1MSTK,POPMTH,AVE=D1,D1FSTK,CHKASN,START
704                 **
705                 ** Uses:
706                 **  Inclusive: A,B,C,D,R0-R4,D0,D1,STMTD0,STMTR0,STMTR1,FUNCxx,
707                 **             ST[11:0],all RAM EXPEXC is permitted to use
708                 **
709                 ** Stk Lvls:   5 (EXPEXC)
710                 **
711                 ** History:
712                 **
713                 **    Date     Programmer            Modification
714                 **  --------   ----------    ---------------------------------
715                 **  12/16/83      NZ         Updated documentation
716                 **                SC         Wrote routine
717                 **
718                 ****************************************************************
719                 ****************************************************************
720 F222B 0               CON(1) =FIXSPC       6 nibbles available here
721 F222C                 BSS    6-1
722                 *-
723                 *-
724 F2231 8F00  =NXTDST  GOSBVL =CHKEOL       Check if EOL yet
          000
725 F2238 500           RTNNC                Yes...return with carry clear
726                 *
```

```
    727 F223B 161          DO=DO+ 2
    728 F223E 7D80         GOSUB  Mflg=0       Clear MLFFLG so can tell if UDF used
    729 F2242 8E00         GOSUBL =eXPEXC      Evaluate the variable
              00
    730 F2248 8F00 NXTDS-  GOSBVL =NXTVA-      Create it, if needed, and set it up
              000
    731 F224F 8F00         GOSBVL =D1MST+      Set D1 to top of stack,clear ST
              000
    732 F2256 8F00         GOSBVL =STKVCT      Set appropriate status bits
              000
    733            *
    734            * Do not allow an array or a complex number as the destination
    735            *
    736 F225D 873          ?ST=1  Cmplex       Is it complex?
    737 F2260 70           GOYES  BADTYP       Yes...Type error
    738 F2262 861          ?ST=0  Array        Is it array?
    739 F2265 91           GOYES  NXTD10       No...continue
    740 F2267 8D00 BADTYP  GOVLNG =RDATTY      Yes...Data Type error
              000
    741            *-
    742            *-
    743 F226E 0            CON(1) =FIXSPC      16 nibbles available here
    744 F226F              BSS    16-1
    745            *-
    746            *-
    747 F227E 7EC3 NXTD10  GOSUB  D1mstk       Reset D1 to top of stack
    748 F2282 8F00         GOSBVL =POPMTH      Pop off the variable value
              000
    749 F2289 7E2F         GOSUB  aVE=D1       Set AVMEME=D1
    750 F228D 7856         GOSUB  D1fstk       Set D1 to (FORSTK)
    751 F2291 1C6          D1=D1- 7            Move to (FORSTK)-7 (Device addr)
    752 F2294 15F6         C=DAT1 7            Read device address & info
    753 F2298 1B00         DO=(5) =MLFFLG      Check if a UDF has been called
              000
    754 F229F 14A          A=DATO B
    755 F22A2 908          ?A=0   P            User-defined function?
    756 F22A5 E1           GOYES  NXTD20       No...continue
    757 F22A7 32FF         LCHEX  FFF          Yes...set device address to search
              F
    758 F22AC 8E00         GOSUBL =CHKASN      Figure out how to find the device
              00
    759 F22B2 D7           D=C    A
    760 F22B4 8E00         GOSUBL =START       Find the device
              00
    761 F22BA 407          GOC    ENTRex       Error setting up the device address
    762 F22BD DB           C=D    A
    763 F22BF 1553         DAT1=C X            Write out the (new) device address
    764            *
    765 F22C3 7983 NXTD20  GOSUB  D1mstk       Position back to top of stack
    766 F22C7 137          CD1EX               Set C[A]=D1 = top of stack
    767 F22CA 135          D1=C
    768 F22CD 02           RTNSC               Return with carry set...good var
    769            *-
    770            *-
    771 F22CF 8F00 Mflg=0  GOSBVL =SVTRC       Save pointer for TRACE
```

```
                000
    772 F22D6 8D00          GOVLNG =MFLG=0       Clear multi-UDF flag
                000
```

```
                STITLE Read characters from loop
*****************************************************************
*****************************************************************
**
** Name:        RED-LF - Read characters from the loop until <Lf>
** Name:        SKP-LF - Read & discard characters from the loop
** Name:        REDCOO - Read characters from the loop until <Lf>
** Name:        REDCHR - Read characters from the loop
** Name:        RDST01 - Read characters from the loop to stack
**
** Category:    LOCAL
**
** Purpose:
**      Read data from the loop onto the stack
**
** Entry:
**      REDCHR,REDCOO,RED-LF,SKP-LF only:
**        The 7 nibble device specifier is stored on the bottom
**          (highest address) of the math stack.
**      RDST01 only:
**        R1[6:0] is the 7-nibble device specifier
**
**      (All entries)
**
**      P=0,HEXMODE
**      D1 points to current top of math stack. Data read will
**        be stored on top of stack (last character placed at
**        lowest address)
**
**      Available memory on stack will be checked.
**
**      S5 (BytCnt):
**           1:Read a specified number of characters
**               A[A] is the number of characters to read
**           0:Terminate by END frame or terminating char match
**               A[B] is the terminating character
**
**      S6 (Trash):
**           1:Ignore the data which is read
**           0:Save the data which is read on the stack
**
**      S7 (ChrTrp):
**           1:Detect a special character in incoming data
**               B[B] is the character to be detected
**               If B[3:2]=00, ignore the character;
**               otherwise replace the character with B[3:2]
**           0:No special character processing
**
**      If system flag -23 is set:
**        Terminate by ETO, terminating character is ignored
**
**        If S5 (BytCnt)=0, S6 (Trash)=0, and S-R0-3[0]>2 (the
**          destination is a string), then S-R1-1[3:0] and R3[A]
**          are the maximum number of chars to read before
**          interrupting the conversation with an NRD.
```

```
828              **          R3[S] must not be "F".
829              **          (R3 is for HPIL:1A only, S-R1-1 for all others)
830              **
831              **        If S5 (BytCnt)=1 or S6 (Trash)=1, then flag -23 has
832              **          no effect other than to terminate on an ETO instead
833              **          of the terminator character.
834              **
835              **        If { S5 (BytCnt)=0 and S-R0-3[0]<=2 (not string dest) }
836              **          OR { device mode (not controller) },
837              **          then flag -23 has no effect (it is ignored).
838              **
839              **
840              ** Exit:
841              **      HEX mode.
842              **      XM=0.
843              **      Carry clear:
844              **        D1 points to the last character read
845              **        Number of chars read=(FORSTK)-D1
846              **        S4 (Menerr)=0
847              **        A[S] contains the state of flag -23 (A[S]=0:flag clear)
848              **      Carry set:
849              **        S4 (Menerr)=1: Insufficient memory (Need to load eMEM)
850              **        S4 (Menerr)=0: P,C[0] is the error code
851              **
852              ** Calls:      FSTK-7,SFLAG?,STGART,CHKSTK,GETDev,CLMODE,CS=TYP,
853              **             PUTC,SETTRM,PUTEFC,YTML,PUTE,GETX,FRAME-,CLMDUT
854              **
855              ** Uses:
856              **  Inclusive: A,B[15:14,A],C,D[15:13,5:0],R1,R2,D0,D1,P,ST[7:0]
857              **
858              ** Stk lvls:   4 (START)
859              **
860              ** History:
861              **
862              **    Date     Programmer            Modification
863              **  --------   ----------   --------------------------------
864              **  01/09/83      NZ        Rewrote character read loop to
865              **                          be faster and shorter
866              **  12/19/83      NZ        Updated documentation
867              **                SC        Wrote routine
868              **
869              ****************************************************************
870              ****************************************************************
871 F22DD 856  SKP-LF  ST=1   Trash         Read and trash data until <Lf>
872 F22E0 6600        GOTO   REDC00
873              *-
874              *-
875 F22E4 846  RED-LF  ST=0   Trash         Keep all data that is read
876 F22E7 845  REDC00  ST=0   BytCnt        Read and save until <Lf>
877 F22EA 847          ST=0   ChrTrp        Don't do special char matching
878 F22ED 1B00         D0=(5) =TERCHR
          000
879 F22F4 14A          A=DAT0 B             Read the terminator char (<Lf>?)
880 F22F7 119  =REDCHR C=R1                 (Preserve the upper nibs of R1)
881 F22FA 72EE         GOSUB  FSTK-7        Get device address from stack...
```

```
    882 F22FE 109          R1=C                ...and save it in R1
    883 F2301 AC0  RDST01  A=0    S            Clear flag -23 indicator nibble
    884 F2304 102          R2=A                Save character count in R2[A]
    885            *
    886            * Save system flag(-23) in R2[S]
    887            *
    888 F2307 3100         LC(2)  =flEOT
    889 F230B 8E00         GOSUBL =sFLAG?      Check if flag -23 is set
              00
    890 F2311 5B0          GONC   RDST05       Not set...leave R2[S]=0
    891 F2314 112          A=R2                Flag -23 is set...set R2[S]
    892 F2317 B44          A=A+1  S
    893 F231A 102          R2=A                Save back in R2
    894 F231D 119  RDST05  C=R1                Recall device address from R1
    895 F2320 D7           D=C    A
    896 F2322 8E00         GOSUBL =START       Set up the mailbox, DO
              00
    897 F2328 560          GONC   RDST10       No error...continue
    898 F232B 664C ENTRex  GOTO   ENTREX       Error...exit
    899            *-
    900            *-
    901 F232F 73F1 RDST10  GOSUB  CHKSTK       Set R1[A] to # bytes available
    902 F2333 DC           ABEX   A            Swap # bytes to B[A], B[3:0] to A
    903 F2335 122          AR2EX               Save B[A] in R2, recall R2 to A[S,A]
    904 F2338 7DB5         GOSUB  getdev       Check if in device mode
    905 F233C 462          GOC    RDST15       Yes...continue
    906 F233F 7B81         GOSUB  CLMODE       No...clear all terminate modes
    907 F2343 47E          GOC    ENTRex       (Error)
    908 F2346 948          ?A=0   S            Is flag -23 clear?
    909 F2349 A1           GOYES  RDST15       Yes...continue
    910 F234B 875          ?ST=1  BytCnt       No...is this by count?
    911 F234E 14           GOYES  RDST25       Yes...continue
    912 F2350 876          ?ST=1  Trash        Not by count...keep data?
    913 F2353 83           GOYES  RDST20       No...set count to "FFFFF"
    914            *
    915            * Keep data which is read, flag -23 is set, not by count
    916            *
    917 F2355 7C9E         GOSUB  CS=TYP       Check if numeric destination
    918 F2359 413          GOC    RDST20       Yes...set byte count to "FFFFF"
    919            *
    920            * System flag -23 is set, destination is a string variable,
    921            * read until EOT received or the string is full.
    922            *
    923 F235C 7231         GOSUB  A=SLEN       Set A[A] to maximum string length
    924            *                           Use the max string length as count
    925 F2360 855          ST=1   BytCnt       (Go to counting mode)
    926            *
    927 F2363 875  RDST15  ?ST=1  BytCnt       Is this a read by count?
    928 F2366 92           GOYES  RDST25       Yes...set it up
    929            *
    930            * Terminate by character matching; always terminate by an END
    931            * frame.  Flag -23 should be ignored for this case.
    932            *
    933 F2368 AC0          A=0    S            Clear flag -23 indicator nibble
    934 F236B 811          BSLC
```

```
    935 F236E 811          BSLC
    936 F2371 AE8          B=A    B            Save the terminator char in B[15:14]
    937 F2374 815          BSRC
    938 F2377 815          BSRC
    939 F237A 3300         LC(4)  (=mSETTM)+12 Set mode to terminate by END frame
              00
    940 F2380 7C51         GOSUB  putc
    941 F2384 46A          GOC    ENTRex       Error
    942 F2387 7181         GOSUB  SETTRM       Set terminate by character match
    943 F238B D0   RDST20  A=0    A            Set byte count to "FFFFF"
    944 F238D CC           A=A-1  A
    945            *
    946 F238F 96B  RDST25  ?D=0   B            Is the device LOOP?
    947 F2392 80           GOYES  RDST30       Yes...leave addressing as it is
    948            *
    949            * All non-controller devices will have D[B]=0!
    950            *
    951 F2394 8E00         GOSUBL =YTML        No...address the device as talker
              00
    952 F239A 8A8  RDST30  ?A=0   A            Is the byte count zero?
    953 F239D E0           GOYES  RDST35       Yes...goto RDST75 (out of range)
    954 F239F D6           C=A    A            No...start conversation
    955 F23A1 8E00         GOSUBL =hCPY5s      Load either SDA or Set frame count
              00
    956 F23A7 7C41         GOSUB  pute         Send data, count=A[A]
    957            *
    958            * Start of main data read loop
    959            *
    960 F23AB 8A8  RDST35  ?A=0   A            Is the count to zero?
    961 F23AE F7           GOYES  RDST75       Yes...exit
    962 F23B0 8E00         GOSUBL =GETX        No...read next message
              00
    963 F23B6 435          GOC    RDST65       Not data...check frame
    964 F23B9 CC   RDST40  A=A-1  A            Decrement count
    965 F23BB 876          ?ST=1  Trash        Is this data to keep?
    966 F23BE E3           GOYES  RDST55       No...process next byte
    967 F23C0 867          ?ST=0  ChrTrp       Is this special char trapping?
    968 F23C3 42           GOYES  RDST50       No...store it
    969            *
    970            * Special character processing
    971            *
    972 F23C5 122          AR2EX               Save count in R2, get chars
    973 F23C8 966          ?A#C   B            Is this the special character?
    974 F23CB 61           GOYES  RDST45       No...restore A, R2; continue
    975 F23CD 814          ASRC                Yes...see what to do
    976 F23D0 814          ASRC
    977 F23D3 AE6          C=A    B            Copy the replace char/delete flag
    978 F23D6 810          ASLC
    979 F23D9 810          ASLC
    980 F23DC 96E          ?C#0   B            Test char to set carry if replace
    981 F23DF 20           GOYES  RDST45       Carry SET to replace,CLEAR to delete
    982 F23E1 122  RDST45  AR2EX               Restore A, R2
    983 F23E4 521          GONC   RDST52       This was delete...ignore it
    984 F23E7 874  RDST50  ?ST=1  Memerr       Has stack collision occurred?
    985 F23EA 21           GOYES  RDST55       Yes...do next char
```

```
 986 F23EC CD          B=B-1  A          No...check if room for this char
 987 F23EE 451         GOC    RDST60     No room...set memerr
 988 F23F1 1C1         D1=D1- 2          Room...decrement stack pointer
 989 F23F4 14D         DAT1=C B          Write out the character
 990 F23F7 BF6  RDST52 CSR    W          Shift to the next character, if any
 991 F23FA F6          CSR    A
 992 F23FC 0D   RDST55 P=P-1              See if any characters left
 993 F23FE 5AB         GONC   RDST40     Yes...process next char
 994 F2401 49A         GOC    RDST35     Go always...get more chars
 995            *-
 996            *-
 997 F2404 854  RDST60 ST=1   Memerr
 998 F2407 44F         GOC    RDST55     Go always
 999            *-
1000            *-
1001 F240A     RDST65
1002            *
1003            * GETX returned in an error condition:
1004            * If an ETO was received and flag -23 is clear, send SDA again
1005            * If an ETO was received and flag -23 is set, finished
1006            * If matched terminating character, finished
1007            *
1008 F240A 890         ?P=    =eABORT    Is this an abort?
1009 F240D 62          GOYES  RDST80     Yes...exit immediately
1010 F240F 8E00        GOSUBL =FRAME-    No...check the frame
          00
1011 F2415 880         ?P#    =pEOT      Is this an EOT?
1012 F2418 B0          GOYES  RDST70     No...check more
1013            *
1014            * EOT received: check if flag -23 is set (to terminate on EOT).
1015            * If it is not set, send an SDA to continue the conversation.
1016            *
1017 F241A 94C         ?A#0   S          Is flag -23 set?
1018 F241D 01          GOYES  RDST75     Yes...exit
1019 F241F 6A7F RDS30. GOTO   RDST30     No...send SDA again
1020            *-
1021            *-
1022 F2423 880  RDST70 ?P#    =pTERM     Is it terminator character match?
1023 F2426 71          GOYES  RDST85     No...unexpected frame
1024            *
1025            * Terminating char was detected.
1026            * If we are in byte count mode, just keep reading until the
1027            *  byte count reaches zero.
1028            *
1029 F2428 875         ?ST=1  BytCnt     Is this a read by byte count?
1030 F242B 4F          GOYES  RDS30.     Yes...keep reading
1031 F242D 20  RDST75  P=     0          No...set P=0, exit
1032 F242F 6330        GOTO   RDST90
1033            *-
1034            *-
1035 F2433 D3  RDST80  D=0    A          Don't send UNT
1036 F2435 7180        GOSUB  CLMDUT     Try to clean up the mailbox
1037 F2439 20          P=     =eABORT    (Ignore any error from CLMDUT)
1038 F243B 02          RTNSC             Set eABORT, set carry for error
1039            *-
```

```
1040                *-
1041 F243D 80F0 RDST85  CPEX   0            Save P in C[0] (could be C=P 0)
1042 F2441 D5           B=C    A            Save the error code in B for now
1043 F2443 7370         GOSUB  CLMDUT       Clear mode, untalk (if possible)
1044 F2447 D9           C=B    A            Restore the error code from B
1045 F2449 80D0         P=C    0            Recall P value for error
1046 F244D 880          ?P#    =pSTATE      Is the error code in the mailbox?
1047 F2450 90           GOYES  RDST87       No...set generic error
1048 F2452 80D4         P=C    4            Yes...read the error code
1049 F2456 540          GONC   RDST89       Go always
1050                *-
1051                *-
1052 F2459 20   RDST87  P=     =eUNEXP      Unexpected frame error
1053 F245B 80F0 RDST89  CPEX   0            Put error code into C[0]
1054 F245F 20           P=     =ePIL        Set P to ePIL error code
1055 F2461 02           RTNSC               Set carry to indicate error exit
1056                *-
1057                *-
1058                *
1059                * End of main data entry loop
1060                *
1061                * The following code is to clean up after normal termination
1062                *
1063 F2463 7350 RDST90  GOSUB  CLMDUT       Clear mode and send UNT
1064 F2467 400          RTNC               (Error)
1065 F246A 811          BSLC
1066 F246D 811          BSLC               B[B] is the terminator character
1067 F2470 874          ?ST=1  Memerr      Was there a stack collision?
1068 F2473 00           RTNYES             Yes...insufficient memory
1069 F2475 875          ?ST=1  BytCnt      Is this a read by count?
1070 F2478 51           GOYES  RDST95       Yes...don't strip "terminator" char
1071 F247A 876          ?ST=1  Trash       Is this read but through away?
1072 F247D 01           GOYES  RDST95       Yes...don't look at garbage!
1073 F247F 853          ST=1   Endfrm      Assume an END frame first
1074 F2482 14F          C=DAT1 B           Check the last character
1075 F2485 965          ?B#C   B           Is it the terminator character?
1076 F2488 80           GOYES  RDST99       No...keep the last character
1077 F248A 171          D1=D1+ 2           Yes...throw away terminator char
1078 F248D 843  RDST95  ST=0   Endfrm      Last frame is not an END frame
1079 F2490 03   RDST99  RTNCC              Clear carry to indicate all OK
```

```
1080                    STITLE Utility routines
1081            ****************************************************************
1082            ****************************************************************
1083            **
1084            ** Name:        A=SLEN - Set A[A] to the string length
1085            **
1086            ** Category:    LOCAL
1087            **
1088            ** Purpose:
1089            **      Read the string length from S-R1-1 into A[A]
1090            **
1091            ** Entry:
1092            **      None
1093            **
1094            ** Exit:
1095            **      A[A] is string length (a[4]=0)
1096            **
1097            ** Calls:       None
1098            **
1099            ** Uses.......
1100            **  Inclusive: A[A]
1101            **
1102            ** Stk lvls:   1 (internal push)
1103            **
1104            ** History:
1105            **
1106            **    Date      Programmer              Modification
1107            **  --------    ----------    ---------------------------------
1108            **  01/12/84      NZ          Wrote routine
1109            **
1110            ****************************************************************
1111            ****************************************************************
1112 F2492 06   A=SLEN  RSTK=C                Save C[A] on RSTK
1113 F2494 137         CD1EX                 Save D1 in C[A]
1114 F2497 1F00        D1=(5) =S-R1-1
            000
1115 F249E D0          A=0    A              Clear A[4]
1116 F24A0 15B3        A=DAT1 4              Read string length
1117 F24A4 137         CD1EX                 Restore D1
1118 F24A7 07          C=RSTK                Restore C[A]
1119 F24A9 01          RTN                   Return (carry unchanged)
1120            *-
1121            *-
1122 F24AB 0           CON(1) =FIXSPC        15 nibbles available here
1123 F24AC             BSS    15-1
1124            ****************************************************************
1125            ****************************************************************
1126            **
1127            ** Name:        CLMDUT - Clear terminator modes, send UNT
1128            ** Name:        CLMODE - Clear terminator modes
1129            **
1130            ** Category:    LOCAL
1131            **
1132            ** Purpose:
1133            **      Clean up any special terminator modes set up by ENTER,
```

```
1134            **      set up default modes:
1135            **        Controller: No terminator modes enabled
1136            **        Device: Terminate on <Lf> or END frame
1137            **
1138            ** Entry:
1139            **      DO points to the mailbox
1140            **      Bit 2 (=Device) of LOOPST indicates whether device or
1141            **        controller
1142            **
1143            ** Exit:
1144            **      Carry clear:
1145            **        P=0
1146            **      Carry set:
1147            **        Error (P, C[0] are the error code)
1148            **
1149            ** Calls:      GETDev,UNT,PUTC
1150            **
1151            ** Uses.......
1152            **  Inclusive: C[W],P,ST[3:0]
1153            **
1154            ** Stk lvls:   1 (GETDev:-1 level saved in C[A])(UNT)(PUTC)
1155            **
1156            ** History:
1157            **
1158            **    Date      Programmer            Modification
1159            **  --------    ----------    --------------------------------
1160            **  12/19/83      NZ          Added documentation
1161            **  04/01/82      SC          Wrote routine
1162            **
1163            ****************************************************************
1164            ****************************************************************
1165 F24BA 07   CLMDUT  C=RSTK               Save 1 RSTK level used by GETDev
1166 F24BC 7934         GOSUB  getdev        Check if we are in device mode
1167 F24C0 06           RSTK=C               Restore the RSTK level
1168 F24C2 4A3          GOC    TER/LF        If in device mode, set frame count=0
1169            *
1170            * Controller
1171            *
1172 F24C5 96B          ?D=0   B             Is the device LOOP?
1173 F24C8 60           GOYES  CLMODE        Yes...don't send an UNT
1174 F24CA 7810         GOSUB  UNT           No...send an UNT
1175 F24CE 20   =CLMODE P=     0
1176 F24D0 3300         LC(4)  =mSETTM       Clear terminate on character match
          00
1177 F24D6 7600         GOSUB  putc
1178 F24DA 3300         LC(4)  (=mSETTM)+8   Clear terminate on END frame
          00
1179 F24E0 8C00 putc    GOLONG =PUTC
          00
1180            *-
1181            *-
1182 F24E6 20   =UNT    P=     0             Send the UNT frame
1183 F24E8 3300         LC(4)  =mUNT
          00
1184 F24EE 61FF         GOTO   putc
```

```
1185                *-
1186                *-
1187                *
1188                * C[A] is the frame count
1189                *
1190 F24F2 25      putefc  P=     5
1191 F24F4 300             LC(1)  =mSFC@5       Load "SET FRAME COUNT" opcode
1192 F24F7 8C00    pute    GOLONG =PUTE
          00
1193                ********************************************************************
1194                ********************************************************************
1195                **
1196                ** Name:       TER/LF - Set up to terminate conversation on <Lf>
1197                ** Name:       SETTRM - Set up to terminate on character in A[B]
1198                **
1199                ** Category:   LOCAL
1200                **
1201                ** Purpose:
1202                **      Enable terminate on character match mode, with the
1203                **      character to match set to <Lf>
1204                **
1205                ** Entry:
1206                **      DO points to the mailbox
1207                **      SETTRM only: A[B] is the terminating character
1208                **
1209                ** Exit:
1210                **      Carry clear:
1211                **        P=0, frame count is zero, terminate on <Lf>
1212                **      Carry set:
1213                **        P, C[0] are the error code
1214                **
1215                ** Calls:      PUTEFC,PUTC
1216                **
1217                ** Uses:
1218                **  Inclusive: A[A],C[A],P,ST[3:0]  (A[A] only for TER/LF)
1219                **
1220                ** Stk lvls: 1 (PUTEFC)(PUTC)
1221                **
1222                ** History:
1223                **
1224                **    Date     Programmer             Modification
1225                **  --------   ----------    ---------------------------------
1226                **  12/19/83      NZ         Updated documentation
1227                **                SC         Wrote routine
1228                **
1229                ********************************************************************
1230                ********************************************************************
1231 F24FD D2     =TER/LF C=0    A              Set frame count to zero
1232 F24FF 7FEF            GOSUB  putefc
1233 F2503 400             RTNC
1234 F2506 31A0            LCHEX  0A             Set up for <Lf> terminator
1235 F250A DA              A=C    A
1236 F250C 3300  SETTRM    LC(4)  (=mSETTM)+1    Enable terminator character match
          00
1237 F2512 7ACF            GOSUB  putc
```

```
1238 F2516 400          RTNC
1239 F2519 3300         LC(4)  =#SETTC
           00
1240 F251F AE6          C=A    B              Set terminator character to A[B]
1241 F2522 6DBF         GOTO   putc
```

```
1242                    STITLE Check # bytes mem available
1243                 ************************************************************
1244                 ************************************************************
1245                 **
1246                 ** Name:       CHKSTK - Check how many bytes available on stack
1247                 **
1248                 ** Category:   LOCAL
1249                 **
1250                 ** Purpose:
1251                 **      Check if the math stack has at least 16 bytes available
1252                 **      and return the actual number of bytes available
1253                 **
1254                 ** Entry:
1255                 **      D1 points to the top of the math stack
1256                 **      S6 (Trash)=0: Do the computation
1257                 **      S6 (Trash)=1: Don't bother with computation...don't care
1258                 **
1259                 ** Exit:
1260                 **      Carry clear:
1261                 **        OK (enough room for at least 16 bytes)
1262                 **        R1[A] is number of bytes past 16 that are available
1263                 **        S4 (Memerr)=0
1264                 **      Carry set:
1265                 **        S4 (Memerr)=1
1266                 **
1267                 ** Calls:      D1=AVS
1268                 **
1269                 ** Uses:
1270                 **  Inclusive: A[W],C[W],R1[A],ST[4]
1271                 **
1272                 ** Stk Lvls:   1 (D1=AVS)
1273                 **
1274                 ** History:
1275                 **
1276                 **    Date     Programmer             Modification
1277                 **  --------   ----------    ---------------------------------
1278                 **  12/19/83      NZ         Updated documentation
1279                 **                SC         Wrote routine
1280                 **
1281                 ************************************************************
1282                 ************************************************************
1283 F2526 854  CHKSTK  ST=1   Memerr        Assume there is no room left
1284 F2529 876          ?ST=1  Trash         Check memory available?
1285 F252C 03           GOYES  CKST10        No...don't care (exit)
1286 F252E 1CF          D1=D1- 16            Yes...compute available memory,
1287 F2531 1CF          D1=D1- 16              leaving a 16 byte leeway
1288 F2534 AF2          C=0    W             (Clear nibble 5 for CSRB)
1289 F2537 137          CD1EX                Get stack pointer into C[A]
1290 F253A 8E00         GOSUBL =D1=AVS
          00
1291 F2540 143          A=DAT1 A             Read AVMEMS into A[A]
1292 F2543 135          D1=C                 Restore D1 (-32)
1293 F2546 17F          D1=D1+ 16
1294 F2549 17F          D1=D1+ 16            Now D1 is restored to entry cond'n
1295 F254C E2           C=C-A  A             Compute available memory size
```

```
1296 F254E 400          RTNC                (If carry, less than 16 bytes)
1297 F2551 81E          CSRB                Convert count to bytes
1298 F2554 111          A=R1                Preserve upper nibbles of R1
1299 F2557 DA           A=C    A
1300 F2559 101          R1=A                Write the count to R1[A]
1301 F255C 844  CKST10  ST=0   Memerr       If here, no error
1302 F255F 03           RTNCC
```

```
1303                    STITLE ENTER USING execution
1304            ****************************************************************
1305            * List of external calls and modules:
1306            *
1307            *   AVE=D1  m/f             Set AvMemEnd = D1.
1308            *
1309            *   COUNTC  MB&USG          Count #symbols in C(A),
1310            *                             for #input chars.
1311            *
1312            *   DCRMNT  MB&USG          Decrement symbol multiplier
1313            *                             (e.g., "5D")
1314            *
1315            *   ENDIMG  MB&USG          Reached end of IMAGE string:
1316            *                             test for more input fields.
1317            *
1318            *   NXTEXP  MB&USG          Fetch next expression. Stores
1319            *                             some registers first, then
1320            *                             calls EXPEXC.
1321            *
1322            *   RCVOFS  MB&USG          Recover offset: read offset
1323            *                             from RAM, compute orginal
1324            *                             address.
1325            *
1326            *   TstEnd  MB&USG          Test input list for EOL, @ or "!".
1327            *
1328            *   USloop  MB&USG          Computes address for looping back
1329            *                            to multiplier (e.g., "5D").
1330            *
1331            ****************************************************************
```

```
1332                EJECT
1333          ****************************************************************
1334          *
1335          * Status bits:
1336          *
1337          sCOUNT  EQU    BytCnt        For ENTSTR: "Count input chars"
1338          sTRASH  EQU    Trash         For ENTSTR: "Read but trash chars"
1339          sIGNOR  EQU    ChrTrp        For ENTSTR: "Ignore special char"
1340          *
1341          ****************************************************************
1342          ****************************************************************
1343          *
1344          *
1345          *--- Image tokens for building expanded IMAGE.
1346          ** 1) Tokens not identifying the end of a numeric field.
1347          **   1a) Tokens not used in backwards search.
1348          *  uSTRPT       String pointer
1349          *  uMULT        |D1| Multiplier
1350          *  uLOOPB       Loop on byte
1351          *  uLOOPS       Loop on string (12 nibs)
1352          *  uIMXCH       Strange execution character.
1353          *
1354          **   1b) Tokens used in backwards search.
1355          *  uOPNNM       Open loop without multiplier
1356          *  uJMP{}       Jump over parenthesis loop pointer (9 nibs)
1357          *  uJMPst       Jump over string pointer (14 nibs)
1358          *  uJMPdl       Jump over unfilled delimiter (8nibs)
1359          *  uIMbck       Poll for backward search handler
1360          *  uIMsta       IMAGE string start (|Dx|-see IMentr)
1361          *  uOPNM-       Open loop with mult, decremented
1362          *  uOPNWM       |E0| Open loop with multiplier (ends in 0!)
1363          *
1364          *++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1365          *+ EndNum       Any value >= this identifies the    +
1366          *+              end of a numeric field (used        +
1367          *+              in execution).                      +
1368          *++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1369          *
1370          ** 2) Tokens identifying the end of a numeric field.
1371          **   2a) Tokens not used in backwards search.
1372          *  uCPLXC       Complex field closed
1373          *  uLOOPP       Loop on parentheses (variable #bytes)
1374          *  uIMend       |F0| IMAGE string end
1375          *
1376          **   2b) Tokens used in backwards search.
1377          *  uRESTP       Restart parse
1378          *  uDELIM       Delimiter
1379          **        Tokens delimiting an output/input field.
1380          *  uHKB^        H,K,B or ^ field
1381          *  uALit        "A" literal field
1382          *  uNUMNn       |F8| Numeric, no float chars, no sign*
1383          *  uNUMNs       |F9| Numeric, no float chars, w/sign*
1384          *  uNUMFn       |FA| Numeric, w/float chars, no sign*
1385          *  uNUMFs       |FB| Numeric, w/float chars, w/sign*
1386          *  uNUMEn       |FC| Numeric; w/Exponent, no sign*
```

```
1387            *  uNUMEs      |FD| Numeric, w/Exponent, w/sign*
1388            *
1389            *  *Note: these numeric delimiters have values that
1390            *         determine the status bit setting in USING execute.
1391            *
1392            ************************************************************
1393            ************************************************************
1394            *
1395            * Register usage:
1396            *    The following registers are used in the ENTER USING
1397            *    execution routines, and must be saved during calls to
1398            *    external routines, such as ENTSTR, STOSUB, EXPEXC
1399            *    and SKP-LF:
1400            *     R0[A] = address of execution symbol
1401            *     R3[A] = program counter
1402            *     S8, S9, S10, S11
1403            *
1404            ************************************************************
```

```
1405                    EJECT
1406                ********************************************************************
1407                ********************************************************************
1408                **
1409                ** Name:        ENTUSG - Execute the ENTER USING statement
1410                **
1411                ** Category:    STEXEC
1412                **
1413                ** Purpose:
1414                **      Execute ENTER USING statement.
1415                **
1416                ** Entry:
1417                **      This is a poll handler in response to the pIMXQT poll.
1418                **      The only necessary conditions are:
1419                **        R0[9-5]= address to begin execution of IMAGE tokens
1420                **        RAM set up at AvMemEnd as specified in MB&USG.
1421                **
1422                ** Exit:
1423                **      Through ENDIMG in mainframe (does NOT return from POLL)
1424                **
1425                ** Calls:       D0=PCA,CSRC5,AS=FTY,MEMBER,FINDA,<ENUFND>,
1426                **              <CHRCNT>,<ENT"X">,<ENTstr>,<ENTmlt>,<ENTlpb>,
1427                **              <ENT"C">,<ENT"P">,<ENT"H">,<ENT"K">,<ENTlps>,
1428                **              <ENTlpp>,<ENTrst>,<ENDend>,<ENTdlm>,<END"B">,
1429                **              <ENT"/">,<ENT"R">,<IMerr>
1430                **
1431                ** Uses:        A-D,R0-R4,D0,D1,STMTDx,FUNCxx,ST[11:0],all
1432                **              RAM that EXPEXC is permitted to use
1433                **
1434                ** Stk lvls:    5 (<ENUFND>)
1435                **
1436                ** NOTE:
1437                **      ENTUSG is the driving routine to execute the IMAGE
1438                **      tokens.  Each token has its own execution routine.
1439                **
1440                ** Detail:
1441                **      Call MEMBER and FINDA to execute each token.
1442                **
1443                ** History:
1444                **
1445                **    Date      Programmer              Modification
1446                **  --------    ----------    ---------------------------------
1447                **  01/10/84       NZ         Updated documentation
1448                **  01/06/83       MB         Wrote routines.
1449                **
1450                ********************************************************************
1451                ********************************************************************
1452 F2561 8F00    =ENTUSG GOSBVL =D0=PCA
       000
1453 F2568 161           D0=D0+ 2              Step over the line length
1454 F256B AFA           A=C    W              Set A[15:6]=C[15:6] for test
1455 F256E 3500          LC(6)  =tENTER
       0000
1456 F2576 15A5          A=DATO 6              Read current instruction
1457 F257A 972           ?A=C   W              Is this ENTER USING?
```

```
1458 F257D 40          GOYES  ENTU00        Yes...process it
1459 F257F 00          RTNSXM                No...return carry clear, XM=1
1460           *-
1461           *-
1462 F2581 118  ENTU00 C=R0                  R0[9-5]=execute address.
1463 F2584 8E00        GOSUBL =CSRC5         Execute address to C[A].
           00
1464 F258A 135         D1=C                  To D1.
1465 F258D 171         D1=D1+ 2              Undo next D1=D1-2.
1466 F2590 738C ENTU05 GOSUB  AS=FTY         Zero input flag
1467           *
1468 F2594 D1   ENTU07 B=0    A              B[A]= counter for input chars.
1469 F2596 1C1  ENTU09 D1=D1- 2              Execute next token.
1470 F2599 14B         A=DAT1 B
1471           *
1472 F259C 3100        LC(2)  =uHKB^         Check if end of field.
1473 F25A0 9E2         ?A<C   B              End field token match?
1474 F25A3 60          GOYES  ENTU20         No...check tokens.
1475 F25A5 68B0        GOTO   ENUFLD         Yes...match end field.
1476           *-
1477           *-
1478 F25A9     ENTU20
1479           *
1480           *       LCASC  \.MS*AZDE\     Following 9 lines do this.
1481           *
1482 F25A9 3F          NIBHEX 3F             Next 8 tokens count input chars.
1483 F25AB 54          CON(2) \E\            Input 5 chars (exponent).
1484 F25AD 44          CON(2) \D\            Input digit
1485 F25AF A5          CON(2) \Z\            Input digit
1486 F25B1 14          CON(2) \A\            Input ASCII char.
1487 F25B3 A2          CON(2) \*\            Input digit
1488 F25B5 35          CON(2) \S\            Input digit
1489 F25B7 D4          CON(2) \M\            Input digit
1490 F25B9 E2          CON(2) \.\            Input digit or "."
1491           *
1492 F25BB 2F          P=     15
1493 F25BD 8F00        GOSBVL =MEMBER        Check if token in A[B] matches
           000
1494 F25C4 460         GOC    ENTU30         No...check for other tokens.
1495 F25C7 6B31        GOTO   CHRCNT         Yes...take care of count.
1496           *-
1497           *-
1498 F25CB 8F00 ENTU30 GOSBVL =FINDA         Execute next token.
           000
1499           *
1500 F25D2 85          CON(2) \X\            Skip input char.
1501 F25D4 891         REL(3) ENT"X"
1502           *
1503 F25D7 00          CON(2) =uSTRPT        Pointer to imbedded literal.
1504 F25D9 681         REL(3) ENTstr           (Skip chars)
1505           *
1506 F25DC 00          CON(2) =uMULT         Multiplier.
1507 F25DE 2A1         REL(3) ENTmlt
1508           *
1509 F25E1 00          CON(2) =uLOOPB        Loop on byte.
```

```
1510 F25E3 7A1          REL(3) ENTlpb
1511              *
1512 F25E6 34           CON(2) \C\          Input char or ignore ",".
1513 F25E8 F31          REL(3) ENT"C"
1514              *
1515 F25EB 05           CON(2) \P\          Input char or ignore ".".
1516 F25ED C31          REL(3) ENT"P"
1517              *
1518 F25F0 84           CON(2) \H\          Input compact form (European).
1519 F25F2 FE1          REL(3) ENT"H"
1520              *
1521 F25F5 B4           CON(2) \K\          Input compact form.
1522 F25F7 CF1          REL(3) ENT"K"
1523              *
1524 F25FA 00           CON(2) =uLOOPS      Loop on string.
1525 F25FC 091          REL(3) ENTlps
1526              *
1527 F25FF 00           CON(2) =uLOOPP      Loop on parentheses.
1528 F2601 D81          REL(3) ENTlpp
1529              *
1530 F2604 00           CON(2) =uRESTP      Restart parse.
1531 F2606 491          REL(3) ENTrst
1532              *
1533 F2609 00           CON(2) =uIMend      IMAGE end.
1534 F260B 0E0          REL(3) ENTend
1535              *
1536 F260E 00           CON(2) =uDELIM      Unfilled deliniter.
1537 F2610 640          REL(3) ENTdlm
1538              *
1539 F2613 24           CON(2) \B\          Input byte form.
1540 F2615 3A1          REL(3) ENT"B"
1541              *
1542 F2618 F2           CON(2) \/\          Read record to EOL.
1543 F261A B81          REL(3) ENT"/"
1544              *
1545 F261D 25           CON(2) \R\          Digit or convert "," to "."
1546 F261F E01          REL(3) ENT"R"
1547              *
1548 F2622 E5           CON(2) \^\          Skip over one variable
1549 F2624 C6F          REL(3) ENTU05
1550              *
1551 F2627 00           CON(2) =uCPLXC      Complex execute
1552 F2629 800          REL(3) =CPLXER        (Error exit)
1553              *
1554              *  These IMAGE symbols are skipped for input:
1555              *
1556              *       @                  (Form Feed)
1557              *       uIMXCH             (Unrecognized IMAGE char)
1558              *
1559 F262C 00           CON(2) 0            Others skip to next token.
1560 F262E 5B2          GONC   ENTa09       Go always.
1561              *-
1562              *-
1563 F2631 8D00 CPLXER  GOVLNG =IMerr
          000
```

```
1564            ****************************************************************
1565            ****************************************************************
1566            **
1567            ** Name:       STRPcr - Strip trailing <Cr>, if any
1568            **
1569            ** Category:   LOCAL
1570            **
1571            ** Purpose:
1572            **      Remove the last character from the string if it is a <Cr>
1573            **
1574            ** Entry:
1575            **      P=0
1576            **      D1 points to the top of the stack (lowest address)
1577            **
1578            ** Exit:
1579            **      D1 adjusted if last character was a <Cr>
1580            **      Carry set if no <Cr>, carry clear if removed <Cr>
1581            **
1582            ** Calls:      None
1583            **
1584            ** Uses.......
1585            **  Inclusive: A[B],C[B],D1
1586            **
1587            ** Stk lvls:   0
1588            **
1589            ** History:
1590            **
1591            **    Date      Programmer              Modification
1592            **  --------    ----------    --------------------------------
1593            **  12/02/83      NZ          Added documentation
1594            **  04/01/82      SC          Wrote routine
1595            **
1596            ****************************************************************
1597            ****************************************************************
1598 F2638 31D0 STRPcr  LCHEX  0D            See if the last char is a <Cr>
1599 F263C 14B          A=DAT1 B
1600 F263F 966          ?A#C   B             Is it a <Cr>?
1601 F2642 00           RTNYES               No...return
1602 F2644 171          D1=D1+ 2             Yes...strip it
1603 F2647 03           RTNCC
1604            *-
1605            *-
1606 F2649 8D00 strhed  GOVLNG =STRHED
           000
1607            *-
1608            *-
1609 F2650 8C00 D1=stk  GOLONG =D1@AVE
           00
```

```
1610                EJECT
1611            ****************************************************************
1612            ****************************************************************
1613            **
1614            ** Name:        ENUFLD - Clean up old field, set up new field
1615            ** Name:        ENTdlm - Clean up old field (reached delimiter)
1616            **
1617            ** Category:    LOCAL
1618            **
1619            ** Purpose:
1620            **      "A new ENTER field has been encountered in the IMAGE"
1621            **      Clean up the old one and prepare for the new.
1622            **
1623            ** Entry:
1624            **      P=0
1625            **      D1 is the current execute pointer
1626            **      B[A] is number of input characters (in DECIMAL)
1627            **      STMTD1 contains current stack pointer
1628            **      The 7 nibble device specifier is on the bottom of MTHSTK
1629            **
1630            ** Exit:
1631            **      P=0
1632            **      D1 is the execute pointer for next item
1633            **      STMTD1 contains current stack pointer
1634            **      Device specifier unchanged on MTHSTK
1635            **
1636            ** Calls:
1637            **   ENTdlm:    STORFL
1638            **   ENUFLD:    STORFL,AS=FTY,TstEnd,Mflg=0,NXTEXP,NXTDS-,SAVED1,
1639            **              RCVOFS,SKP-LF,<ENTU07>,<RTNCHK>
1640            **
1641            ** Uses:        A,B,C,D,R0-R4,D0,D1,P,ST[11:0],STMTxx,FUNCxx,
1642            **              All RAM EXPEXC is permitted to use
1643            **
1644            ** Stk lvls:    7 (STORFL)
1645            **
1646            ** Algorithm:
1647            **      Clean up old field:
1648            **        Read in pending chars and store in dest (STORFL)
1649            **      If unfilled delimiter (ENTdlm), then back to ENTUSG.
1650            **      Else (ENUFLD) a new input field is required;
1651            **        Prepare for new field:
1652            **          Save status bits in RAM.
1653            **          Save offset to IMAGE execution in RAM.
1654            **          Check if any more input items:
1655            **            If not, then exit to NXTSTM.
1656            **          Call EXPEXC.  (and DEST via NXTVA-)
1657            **          Restore status bits.
1658            **          Recover offset to IMAGE execution address.
1659            **          Back to ENTUSG.
1660            **
1661            ** History:
1662            **
1663            **    Date      Programmer              Modification
1664            **  --------    ----------    --------------------------------
```

```
1665                ** 01/11/84      NZ          Updated documentation
1666                ** 01/06/83      MB          Wrote routines.
1667                **
1668                ****************************************************************
1669                ****************************************************************
1670 F2656         ENTdlm                         New delimiter, but no enter field.
1671 F2656 76E1           GOSUB  STORFL           Input pending chars, store in dest.
1672 F265A 6B3F ENTa09    GOTO   ENTU09           Next execution symbol.
1673                *-
1674                *-
1675 F265E         ENUFLD                         New enter field.
1676 F265E 7ED1           GOSUB  STORFL           Store previous field.
1677                *
1678                * Save the IMAGE type to CHN#SV (type is in C[S] now)
1679                *  3 - H or K IMAGE
1680                *  2 - String IMAGE
1681                *  1 - Numeric IMAGE
1682                *
1683 F2662 71BB           GOSUB  AS=FTY           Position DO to CHN#SV
1684 F2666 148          A=DAT1 B
1685 F2669 3100         LC(2)  =uHKB^
1686 F266D B46          C=C+1  S                C[S] = 1
1687                *
1688 F2670 962          ?A=C   B                H or K IMAGE? (C[B] = uHKB^)
1689 F2673 31           GOYES  HorK             Yes...set type = 3
1690 F2675 E6           C=C+1  A                (C[B] = uALit)
1691 F2677 962          ?A=C   B                String IMAGE?
1692 F267A F0           GOYES  StrImg           Yes...set type = 2
1693 F267C 5F0          GONC   NumImg           Go always...set type = 1
1694                *-
1695                *-
1696 F267F 8D00 Tstend  GOVLNG =TstEnd
            000
1697                *-
1698                *-
1699 F2686 B46  HorK    C=C+1  S
1700 F2689 B46  StrImg  C=C+1  S
1701 F268C 1544 NumImg  DAT0=C S                Save the IMAGE type in CHN#SV
1702                *
1703 F2690 7BEF           GOSUB  Tstend          Test for end of ENTER stmt.
1704 F2694 564          GONC   EndENT           Yes, end of ENTER stmt.
1705 F2697 133          AD1EX                  Save D1 in A[A] (stack pointer).
1706 F269A 713C         GOSUB  Mflg=0          Clear multi-UDF flag, set TRACE ptr.
1707 F269E 131          D1=A                   Restore D1 from A[A].
1708 F26A1 8F00         GOSBVL =NXTEXP         Get next expression.
            000
1709                *
1710 F26A8 7C9B           GOSUB  NXTDS-          Set up next destination variable.
1711                *
1712                * Get saved PC back from STMTDO and save AVMEME in STMTDO
1713                *
1714 F26AC 1B00         DO=(5) =STMTDO
            000
1715 F26B3 142          A=DAT0 A               A[A] = saved PC
1716 F26B6 103          R3=A                   Save PC in R3
```

```
1717 F26B9 144          DAT0=C A            C[A] = AVMEME from NXTDS-
1718            *
1719 F26BC 8E00         GOSUBL =SAVED1      Save stack pointer in STMTD1
           00
1720            *
1721 F26C2 174          D1=D1+ 5            Set D1 to status storage.
1722 F26C5 147          C=DAT1 A            Read status bits.
1723 F26C8 0A           ST=C                Restore status bits.
1724 F26CA 8F00         GOSBVL =RCVOFS      Recover offset to xqt address.
           000
1725 F26D1 135          D1=C                Position D1 to xqt address.
1726 F26D4 1C7          D1=D1- 8            Skip (unused) field digit counters.
1727 F26D7 6CBE         GOTO   ENTU07       Next execution symbol.
1728            *-
1729            *-
1730 F26DB      EndENT                      End ENTER statement.
1731            *
1732            * The following test must be such that the jump to "exit" has
1733            * carry CLEAR, as RTNCHK checks carry to see if an error has
1734            * occurred.
1735            *
1736 F26DB 966          ?A#C   B            Is it an '#'?
1737 F26DE 50           GOYES  getEOL       No...just read and skip to EOL
1738 F26E0 560          GONC   exit         Go always...just exit
1739            *-
1740            *-
1741 F26E3 76FB getEOL  GOSUB  SKP-LF       Skip characters to EOL.
1742 F26E7 6098 exit    GOTO   RTNCHK       To next statement.
```

```
1743                EJECT
1744          ****************************************************************
1745          ****************************************************************
1746          **
1747          ** Name:       ENTend - Execute the uIMend token
1748          **
1749          ** Category:   LOCAL
1750          **
1751          ** Purpose:
1752          **      Execute the uIMend token.
1753          **
1754          ** Entry:
1755          **      P=0
1756          **      D1 is the current execute pointer
1757          **      B[A] is number of input characters (in DECIMAL)
1758          **      STMTD1 contains current stack pointer
1759          **      The 7 nibble device specifier is on the bottom of MTHSTK
1760          **      From ENTUSG through FINDA.
1761          **
1762          ** Exit:
1763          **      If there are more input items:
1764          **       (returns through ENTU09)
1765          **        P=0
1766          **        D1 is the execute pointer for next item
1767          **        STMTD1 contains current stack pointer
1768          **        Device specifier unchanged on MTHSTK
1769          **      If there are no more input items, exits via EndENT.
1770          **
1771          ** Calls:      STORFL,TstEnd,ENDIMG,<ENTU09>
1772          **
1773          ** Uses:
1774          **  Inclusive: A,B,C,D,R0-R2,R3[15:5],R4,D0,D1,P,RESREG,FUNCD1,
1775          **             ST[11:8,6,5,3:0]
1776          **
1777          ** Stk lvls:   6 (CNTSTR)(<STOSUB>)
1778          **
1779          ** Algorithm:
1780          **      Clean up old field:
1781          **        Read in pending chars and store in dest (STORFL)
1782          **      Restore status bits from RAM at AvMemEnd.
1783          **      Recover offset to beginning of IMAGE string.
1784          **      If input fields have not been found, then
1785          **        "Invalid USING" error (prevents infinite loop
1786          **        when looking for input field).
1787          **      If input field has been found, loop back to
1788          **        recycle IMAGE string.
1789          **
1790          **
1791          ** History:
1792          **    Date     Programmer              Modification
1793          **  --------   ----------    --------------------------------
1794          **  01/11/84      NZ         Updated documentation
1795          **  01/06/83      MB         Wrote routines.
1796          **
1797          ****************************************************************
```

```
1798            ****************************************************************
1799 F26EB       ENTend                          End of IMAGE string.
1800 F26EB 7151          GOSUB  STORFL          Read pending chars, store in dest.
1801 F26EF 7C8F          GOSUB  Tstend          Test end of ENTER stmt.
1802 F26F3 57E           GONC   EndENT          Yes, end of stmt.
1803 F26F6 8F00          GOSBVL =ENDIMG         Test valid flds, D1=start of IMAGE.
           000
1804 F26FD 135           D1=C
1805 F2700 560           GONC   ENTb09          Go always...recycle IMAGE string.
```

```
1806                    EJECT
1807            ****************************************************************
1808            ****************************************************************
1809            **
1810            ** Name:       CHRCNT - Count the number of chars to be input
1811            **
1812            ** Category:   LOCAL
1813            **
1814            ** Purpose:
1815            **      Count the number of chars to be input from loop.
1816            **
1817            ** Entry:
1818            **      P=0
1819            **      D1 is the current execute pointer
1820            **      B[A] is number of input characters (in DECIMAL)
1821            **      STMTD1 contains current stack pointer
1822            **      The 7 nibble device specifier is on the bottom of MTHSTK
1823            **
1824            ** Exit:
1825            **      P=0
1826            **      B[A] is the resultant count
1827            **      D1 is the execute pointer for next item
1828            **      STMTD1 contains current stack pointer
1829            **      Device specifier unchanged on MTHSTK
1830            **
1831            ** Calls:      COUNT
1832            **
1833            ** Uses:
1834            **  Inclusive: A[A],B[A],C[A],D[A],P,D1
1835            **
1836            ** Stk lvls:   3 (COUNT)
1837            **
1838            ** Note:
1839            **      The E symbol generates a tokenized field
1840            **      which looks like this:  "ESZZZ".  So it will
1841            **      always generate 5 digit counts.
1842            **
1843            ** Algorithm:
1844            **      Call COUNTC, which does:
1845            **        If accompanying multiplier (CNTMLT),
1846            **          then set C[A]= multiplier, restore counter.
1847            **          ELSE, set C[A]= 00001.
1848            **        Add C to B (Dec mode).
1849            **      If accompanying multiplier,
1850            **        then restore the count (at D1 + 4) to value
1851            **      Exit to ENTU09.
1852            **
1853            ** History:
1854            **
1855            **    Date     Programmer              Modification
1856            **  --------   ----------    ------------------------------
1857            **  01/11/84       NZ        Updated documentation
1858            **  01/06/83       MB        Wrote routines.
1859            **
1860            ****************************************************************
```

```
1861            ****************************************************************
1862 F2703 7400 CHRCNT  GOSUB  COUNT         Count multiplier, if there.
1863 F2707 6E8E ENTb09  GOTO   ENTU09        Process next execution symbol.
1864            *-
1865            *-
1866 F270B 8F00 COUNT   GOSBVL =COUNTC       Process the count
           000
1867 F2712 04           SETHEX
1868 F2714 890          ?P=    0             Was a count specified?
1869 F2717 00           RTNYES               No...just return
1870 F2719 20           P=     0
1871 F271B 173          D1=D1+ 4
1872 F271E 15D3         DAT1=C 4             Restore the count field to initial.
1873 F2722 1C3          D1=D1- 4
1874 F2725 01           RTN
```

```
1875                    EJECT
1876                ****************************************************************
1877                ****************************************************************
1878                **
1879                ** Name:        ENT"C" - Execute the "C" symbol
1880                ** Name:        ENT"P" - Execute the "P" symbol
1881                ** Name:        ENT"R" - Execute the "R" symbol
1882                **
1883                ** Category:    LOCAL
1884                **
1885                ** Purpose:
1886                **      Execute the "C", "P", and "R" symbols.
1887                **
1888                ** Entry:
1889                **      P=0
1890                **      D1 is the current execute pointer
1891                **      B[A] is number of input characters (in DECIMAL)
1892                **      STMTD1 contains current stack pointer
1893                **      The 7 nibble device specifier is on the bottom of MTHSTK
1894                **
1895                ** Exit:
1896                **      P=0
1897                **      D1 is the execute pointer for next item
1898                **      STMTD1 contains current stack pointer
1899                **      Device specifier unchanged on MTHSTK
1900                **      Exit to ENTU09.
1901                **
1902                ** Calls:       CSRC5,CNTSTR,CSLC5,ENTSTr
1903                **
1904                ** Uses.......
1905                **  Inclusive: A,B,C,D[15:13,5:0],R0-R2,D0,D1,P,ST[7:0],STMTD1
1906                **
1907                ** Stk lvls:   6 (CNTSTR)(ENTSTr)
1908                **
1909                ** Algorithm:
1910                **      For "C": load  002C  (0 byte and ",") into C reg.
1911                **      For "P": load  002E  (0 byte and ".") into C reg.
1912                **      For "R": load  2E2C  ("." and ",") into C reg.
1913                **      Save in R1[15:12].
1914                **      Input all pending chars.
1915                **      Put R1[15:12] in B[3:0].
1916                **      Input one char, ignoring or replacing as specified.
1917                **      Exit to ENTUSG.
1918                **
1919                ** History:
1920                **    Date      Programmer              Modification
1921                **   --------   ----------    ---------------------------------
1922                **   01/11/84      NZ         Updated documentation
1923                **   01/06/83      MB         Wrote routines.
1924                **
1925                ****************************************************************
1926                ****************************************************************
1927 F2727 2C   ENT"C"  P=     12              Loads "," into C[B], 00 in C[3:2].
1928 F2729 0D   ENT"P"  P=P-1                  (For ENT"P", P is 0, gives P=14).
1929 F272B 0D          P=P-1                  Loads "." into C[B], 00 in C[3:2].
```

```
1930 F272D 39C2 ENT"R"  LCHEX  002C002E2C   C[B]= Character to ignore.
           E200
           C200
1931 F2739 8E00         GOSUBL =CSRC5
           00
1932 F273F 109          R1=C                Store character info in R1[15:12].
1933 F2742 7151         GOSUB  CNTSTR       Input pending chars.
1934 F2746 119          C=R1                Char to ignore from R1[15:12]...
1935 F2749 8E00         GOSUBL =CSLC5
           00
1936 F274F D5           B=C    A            ...to B[3:0].
1937 F2751 D0           A=0    A
1938 F2753 E4           A=A+1  A            Input one char.
1939 F2755 857          ST=1   sIGNOR       "Ignore char."
1940 F2758 7B41         GOSUB  ENTSTr       Go read the character.
1941 F275C 5AA          GONC   ENTb09       Go always (next execution symbol).
```

```
1942                    EJECT
1943            ****************************************************************
1944            ****************************************************************
1945            **
1946            ** Name:        ENTstr - Execute the uSTRPT token (string IMAGE)
1947            ** Name:        ENT"X" - Execute the "X" token (skip character)
1948            **
1949            ** Category:    LOCAL
1950            **
1951            ** Purpose:     ENTstr: Execute uSTRPT token.
1952            **              ENT"X": Execute "X" symbol.
1953            **
1954            ** Entry:
1955            **      P=0
1956            **      D1 is the current execute pointer
1957            **      B[A] is number of input characters (in DECIMAL)
1958            **      STMTD1 contains current stack pointer
1959            **      The 7 nibble device specifier is on the bottom of MTHSTK
1960            **
1961            ** Exit:
1962            **      P=0
1963            **      D1 is the execute pointer for next item
1964            **      STMTD1 contains current stack pointer
1965            **      Device specifier unchanged on MTHSTK
1966            **      Exits to ENTU09.
1967            **
1968            ** Calls:       CNTSTR,COUNT,CNTST1
1969            **
1970            ** Uses.......
1971            **  Inclusive: A,B,C,D[15:13,5:0],R0-R2,D0,D1,P,STMTD1,ST[7:0]
1972            **
1973            ** Stk lvls:    6 (CNTSTR)(CNTST1)
1974            **
1975            ** Algorithm:
1976            **      ENTstr:  Input pending chars.
1977            **               Read in length of literal = #chars to trash
1978            **               Goto ENTX07.
1979            **      ENT"X":  Input pending chars.
1980            **               If accompanied by multiplier, read multiplier
1981            **                 into C[A].  Else, set C[A]=1.
1982            **         ENTX07 Read in specified #chars and trash.
1983            **               Exit to ENTU09.
1984            **
1985            ** History:
1986            **    Date      Programmer              Modification
1987            **  --------    ----------    ---------------------------------
1988            **  01/11/84      NZ          Updated documentation
1989            **  01/06/83      MB          Wrote routines.
1990            ****************************************************************
1991            ****************************************************************
1992 F275F     ENTstr                        IMAGE Literal.
1993 F275F 7431         GOSUB  CNTSTR        Input necessary chars.
1994 F2763 1C9          D1=D1- 10            To literal length.
1995 F2766 147          C=DAT1 A             Literal length= #chars to trash.
1996 F2769 5A0          GONC   ENTX07        Go always...read and trash chars.
```

```
1997            *-
1998            *-
1999 F276C 7721 ENT"X"  GOSUB  CNTSTR      Input necessary chars.
2000 F2770 779F         GOSUB  COUNT       Count multiplier, if there.
2001 F2774 D5   ENTX07  B=C    A           Put count into B[A].
2002 F2776 856          ST=1   sTRASH      "Read but trash chars."
2003 F2779 7D11         GOSUB  CNTST1      Input chars.
2004 F277D 598  ENTc09  GONC   ENTb09      Go always...execute next symbol.
```

```
2005                    EJECT
2006            ****************************************************************
2007            ****************************************************************
2008            **
2009            ** Name:       ENTMLT - Execute the uMULT token.
2010            **
2011            ** Category:   LOCAL
2012            **
2013            ** Purpose:
2014            **      Execute the uMULT token.
2015            **
2016            ** Entry:
2017            **      P=0
2018            **      D1 is the current execute pointer
2019            **      B[A] is number of input characters (in DECIMAL)
2020            **      STMTD1 contains current stack pointer
2021            **      The 7 nibble device specifier is on the bottom of MTHSTK
2022            **
2023            ** Exit:
2024            **      P=0
2025            **      D1 is the execute pointer for next item
2026            **      STMTD1 contains current stack pointer
2027            **      Device specifier unchanged on MTHSTK
2028            **
2029            ** Calls:      DCRMNT
2030            **
2031            ** Uses.......
2032            **  Inclusive: A[B],C[A],D1
2033            **
2034            ** Stk lvls:   1 (DCRMNT)
2035            **
2036            ** Algorithm:
2037            **      Move D1 to multiplier reserve, check if open
2038            **        parentheses loop  (uOPNWM).
2039            **      If it is, change uOPNWM to uOPNM-.
2040            **      Move D1 to mulitplier counter, decrement.
2041            **      If no carry, exit to ENTUSG.
2042            **      If carry, restore counter to reserve value,
2043            **        set D1= value saved in D(A), exit to ENTUSG.
2044            **
2045            ** History:
2046            **    Date      Programmer              Modification
2047            **  --------    ----------    --------------------------------
2048            **  01/11/84       NZ         Updated documentation.
2049            **  01/06/83       MB         Wrote routines.
2050            **
2051            ****************************************************************
2052            ****************************************************************
2053 F2780 8F00 ENTmlt  GOSBVL =DCRMNT        Decrement multiplier.
         000
2054 F2787 55F          GONC   ENTc09         Go always...next execution symbol.
```

```
2055                    EJECT
2056                ****************************************************************
2057                ****************************************************************
2058                **
2059                ** Name:        ENTlpb - Execute the uLOOPB token
2060                ** Name:        ENTlps - Execute the uLOOPS token
2061                ** Name:        ENTlpp - Execute the uLOOPP token
2062                **
2063                ** Category:    LOCAL
2064                **
2065                ** Purpose:
2066                **      Execute the three loop tokens.
2067                **
2068                ** Entry:
2069                **      P=0
2070                **      D1 is the current execute pointer
2071                **      B[A] is number of input characters (in DECIMAL)
2072                **      STMTD1 contains current stack pointer
2073                **      The 7 nibble device specifier is on the bottom of MTHSTK
2074                **
2075                ** Exit:
2076                **      P=0
2077                **      D1 is the execute pointer for next item
2078                **      STMTD1 contains current stack pointer
2079                **      Device specifier unchanged on MTHSTK
2080                **
2081                ** Calls:       USloop
2082                **
2083                ** Uses:......
2084                **  Inclusive: A[S],C[A],D[A],D1,P
2085                **
2086                ** Stk lvls:    1 (USloop)
2087                **
2088                ** Algorithm:
2089                **      For uLOOPB:  Set P=3
2090                **      For uLOOPS:  Set P=15
2091                **                   Move D1 back to multiplier counter (C+P+1)
2092                **           ENlop3  Save original D1 in D (execution address
2093                **                     in case multiplier decrements past 0.)
2094                **                   Jump to ENm105 to decrmnt counter, etc.
2095                **                      (exits to ENTUSG).
2096                **      For uLOOPP:  Move D1 to offset for open paren.
2097                **                   Recover offset (point to open paren).
2098                **                   Goto ENlop3.
2099                **
2100                ** History:
2101                **    Date      Programmer             Modification
2102                **  --------    ----------   ---------------------------------
2103                **  01/06/83    M. Banwarth  Wrote routines.
2104                **
2105                ****************************************************************
2106                ****************************************************************
2107 F278A 24    ENTlpb  P=     4             (For P=3: back up D1 4 nibs)
2108 F278C 0D    ENTlps  P=P-1                (P=15: back up D1 16 nibs)
2109 F278E 8F00 ENTlpp  GOSBVL =USloop        Back up D1 to multiplier.
```

```
             000
  2110 F2795 20          P=     0
  2111 F2797 55E         GONC   ENTc09        Go always...next execution char.
```

```
2112                    EJECT
2113                ****************************************************************
2114                ****************************************************************
2115                **
2116                ** Name:        ENTrst - Execute the uRESTP token
2117                **
2118                ** Category:    LOCAL
2119                **
2120                ** Purpose:
2121                **      Execute the uRESTP token.
2122                **
2123                ** Entry:
2124                **      P=0
2125                **      D1 is the current execute pointer
2126                **      B[A] is number of input characters (in DECIMAL)
2127                **      STMTD1 contains current stack pointer
2128                **      The 7 nibble device specifier is on the bottom of MTHSTK
2129                **
2130                ** Exit:
2131                **      P=0
2132                **      D1 is the execute pointer for next item
2133                **      STMTD1 contains current stack pointer
2134                **      Device specifier unchanged on MTHSTK
2135                **
2136                ** Uses.......
2137                **  Inclusive: A,B,C,D,R1,R2,R3[15:5],R4,D0,D1,P,RESREG,FUNCD1,
2138                **             ST[11:8,6,5,3:0]
2139                **
2140                ** Stk lvls:   7 (STORFL)
2141                **
2142                ** Algorithm:
2143                **      Store D1 (address of uRESTP token) in R0(9-5)
2144                **      for use back in MB&USG module.  Return from
2145                **      poll with carry cleared, XM=0.
2146                **
2147                ** History:
2148                **    Date      Programmer              Modification
2149                **  --------    ----------    --------------------------------
2150                **  01/11/84       NZ         Updated documentation
2151                **  01/06/83       MB         Wrote routines.
2152                **
2153                ****************************************************************
2154                ****************************************************************
2155 F279A         ENTrst                          Restart parse.
2156 F279A 72A0           GOSUB  STORFL
2157 F279E 8D00           GOVLNG =USGrst          End poll handler.
           000
```

```
2158                    EJECT
2159                ****************************************************************
2160                ****************************************************************
2161                **
2162                ** Name:        ENT"/" - Execute the "/" token
2163                **
2164                ** Category:    LOCAL
2165                **
2166                ** Purpose:     Execute "/" symbol.
2167                **
2168                ** Entry:
2169                **      P=0
2170                **      D1 is the current execute pointer
2171                **      B[A] is number of input characters (in DECIMAL)
2172                **      STMTD1 contains current stack pointer
2173                **      The 7 nibble device specifier is on the bottom of MTHSTK
2174                **
2175                ** Exit:
2176                **      P=0
2177                **      STMTD1 contains current stack pointer
2178                **      Device specifier unchanged on MTHSTK
2179                **
2180                ** Calls:       STORFL,SKP-LF
2181                **
2182                ** Uses:......
2183                **  Inclusive: A,B,C,D,R1,R2,R3[15:5],R4,D0,P,RESREG,FUNCD1,
2184                **             ST[11:0]
2185                **
2186                ** Stk lvls:   7 (STORFL)
2187                **
2188                ** Algorithm:
2189                **      Calls SKIP to skip to EOL of input record.
2190                **
2191                ** History:
2192                **    Date      Programmer              Modification
2193                **  --------    ----------    ---------------------------------
2194                **  01/11/84       NZ         Updated documentation
2195                **  01/06/83       MB         Wrote routines.
2196                **
2197                ****************************************************************
2198                ****************************************************************
2199 F27A5         ENT"/"                          Skip to EOL.
2200 F27A5 7790           GOSUB   STORFL          Store pending item.
2201 F27A9 703B           GOSUB   SKP-LF          Skip to end of line.
2202 F27AD 560            GONC    ENT/03          Go if no error.
2203 F27B0 6A7B           GOTO    ENTRex          Error exit
2204                *-
2205                *-
2206 F27B4 6FDD ENT/03    GOTO    ENTU07          Next execution symbol.
```

```
2207                    EJECT
2208            ****************************************************************
2209            ****************************************************************
2210            **
2211            ** Name:        ENT"B" - Execute the "B" token
2212            **
2213            ** Category:    LOCAL
2214            **
2215            ** Purpose:
2216            **      Execute the "B" symbol.
2217            **
2218            ** Entry:
2219            **      P=0
2220            **      D1 is the current execute pointer
2221            **      B[A] is number of input characters (in DECIMAL)
2222            **      STMTD1 contains current stack pointer
2223            **      The 7 nibble device specifier is on the bottom of MTHSTK
2224            **
2225            ** Exit:
2226            **      P=0
2227            **      D1 is the execute pointer for next item
2228            **      STMTD1 contains current stack pointer
2229            **      Device specifier unchanged on MTHSTK
2230            **
2231            ** Calls:       CNTSTR,STOBIN
2232            **
2233            ** Uses.......
2234            **  Inclusive: A,B,C,D,R0[15:5],R1,R2,R3[15:5],R4,D0,D1,P,
2235            **             RESREG,STMTD1,ST[11:0]
2236            **
2237            ** Stk lvls:    6 (CNTSTR)(<STOBIN>)
2238            **
2239            ** Algorithm:
2240            **      Set B[A]=1 (counter for #chars to input)
2241            **      Read one char (CNTSTR)
2242            **      Exit to ENTU09.
2243            **
2244            ** History:
2245            **    Date      Programmer          Modification
2246            **  --------    ----------    --------------------------------
2247            **  01/12/84       NZ         Updated documentation
2248            **  01/06/83       MB         Wrote routines.
2249            **
2250            ****************************************************************
2251            ****************************************************************
2252 F27B8      ENT"B"                        B field in IMAGE.
2253 F27B8 E5           B=B+1  A              Input one char.  (B[A]=0 already)
2254 F27BA 79D0         GOSUB  CNTSTR         Read the character.
2255 F27BE 146          C=DAT0 A              Read stack pointer (from STMTD1).
2256 F27C1 7400         GOSUB  STOBIN         Convert and store the binary number.
2257 F27C5 6ACD ENTU05  GOTO   ENTU05         Next token.
2258            *-
2259            *-
2260 F27C9 135  STOBIN  D1=C                  Set D1 to top of stack.
2261 F27CC D0           A=0    A
```

```
2262 F27CE 14B          A=DAT1 B            Read the character.
2263 F27D1 8F00         GOSBVL =HDFLT       Convert to floating number.
           000
2264 F27D8 04           SETHEX
2265 F27DA 1CD          D1=D1- 14
2266 F27DD 61B0         GOTO   STODE1       Write value to stack, store it.
```

```
2267                EJECT
2268            ****************************************************************
2269            ****************************************************************
2270            **
2271            ** Name:        ENT"K" - Execute the "K" token
2272            ** Name:        ENT"H" - Execute the "H" token
2273            **
2274            ** Category:    LOCAL
2275            **
2276            ** Purpose:
2277            **      Execute the "K" or "H" token: free format ENTER
2278            **
2279            ** Entry:
2280            **      P=0
2281            **      D1 is the current execute pointer
2282            **      B[A] is number of input characters (in DECIMAL)
2283            **      STMTD1 contains current stack pointer
2284            **      The 7 nibble device specifier is on the bottom of MTHSTK
2285            **
2286            ** Exit:
2287            **      P=0
2288            **      D1 is the execute pointer for next item
2289            **      STMTD1 contains current stack pointer
2290            **      Device specifier unchanged on MTHSTK
2291            **      Exit to ENTU05.
2292            **
2293            ** Calls:       CS=TYP,ENTST2,ENT160,D1@AVE,TstEnd,AVE=D1,ENTST3
2294            **
2295            ** Uses:
2296            **  Inclusive: A,B,C,D,R0,R1,R2,R3[15:5],R4,D0,D1,P,ST[11:0],
2297            **             RESREG
2298            **
2299            ** Stk lvls:   7 (ENT160)
2300            **
2301            ** Algorithm:
2302            **      If destination is numeric and "H",
2303            **        then set up to replace commas with decimal points.
2304            **      Set up to read until terminator character match.
2305            **      Read the data.
2306            **      Do the assignment.
2307            **      If more variables remaining,
2308            **        then set AVMEME back to original value; goto ENTU05.
2309            **        else exit to next statement.
2310            **
2311            ** History:
2312            **
2313            **    Date      Programmer            Modification
2314            **   --------   ----------    --------------------------------
2315            **  12/21/83       NZ         Added GOSUB to ENT"H", changed
2316            **                            GOTO ENTFRM to GOC ENTFRM to fix
2317            **                            SR #0039-1073(6) (ENTER USING "H";A$
2318            **                            with a comma in the input character
2319            **                            sequence)
2320            **                 SC/MB      Wrote
2321            **
```

```
2322            ****************************************************************
2323            ****************************************************************
2324 F27E1 701A ENT"H"  GOSUB  CS=TYP        Check if the destination is string
2325 F27E5 33C2         LCASC  \.,\          Set up to replace "," with "."
           E2
2326 F27EB D5           B=C    A
2327 F27ED 857          ST=1   sIGNOR
2328 F27F0 450          GOC    ENTFFM        Numeric destination...DO change ","
2329            *
2330 F27F3 847  ENT"K"  ST=0   sIGNOR        Don't change commas to "."
2331 F27F6 1B00 ENTFFM  DO=(5) =TERCHR       Read terminating char
           000
2332 F27FD 14A          A=DAT0 B             A[A] is the terminating character.
2333 F2800 845          ST=0   sCOUNT        Don't count chars.
2334 F2803 846          ST=0   sTRASH        Do keep chars.
2335 F2806 70A0         GOSUB  ENTST2        Read the characters.
2336 F280A 146          C=DAT0 A             Recall stack pointer from STMTD1.
2337 F280D 135          D1=C
2338 F2810 855          ST=1   KorH          This is either "K" or "H".
2339 F2813 8EFC         GOSUBL ENT160        Do the assignment
           7F
2340 F2819 733E         GOSUB  D1mstk        Set D1 to AVMEME.
2341 F281D 7E5E         GOSUB  Tstend        Reached end of statement?
2342 F2821 460          GOC    H&Kcnt        No...set up for next item.
2343 F2824 62CE         GOTO   exit          Yes...exit from ENTER USING.
2344            *-
2345            *-
2346 F2828 1B00 H&Kcnt  DO=(5) =STMTD0       Restore AVMEME to its old value
           000
2347 F282F 146          C=DAT0 A             (value was saved by STORFL)
2348 F2832 137          CD1EX
2349 F2835 7289         GOSUB  aVE=D1
2350            *
2351 F2839 7290         GOSUB  ENTST3        Restore D1 to execute address
2352 F283D 578          GONC   ENTu05        Go always (process next item).
```

```
2353                    EJECT
2354                ****************************************************************
2355                ****************************************************************
2356                **
2357                ** Name:        STORFL - Read pending chars, store in destination
2358                **
2359                ** Category:    LOCAL
2360                **
2361                ** Purpose:
2362                **      Read pending input chars, store in dest.
2363                **
2364                ** Entry:
2365                **      P=0
2366                **      D1 is the current execute pointer
2367                **      B[A] is number of input characters (in DECIMAL)
2368                **      STMTD1 contains current stack pointer
2369                **      The 7 nibble device specifier is on the bottom of MTHSTK
2370                **
2371                ** Exit:
2372                **      P=0
2373                **      D1 is the execute pointer for next item
2374                **      STMTD1 contains current stack pointer
2375                **      Device specifier unchanged on MTHSTK
2376                **      B[A]=0
2377                **
2378                ** Calls:       CNTSTR,AS=FTY,CS=TYP,STRHED,GETNUM,POPSTK,D1@AVE,
2379                **              <STOSUB>
2380                **
2381                ** Uses.......
2382                **  Inclusive: A,B,C<D,R0,R1,R2,R3[15:5],R4,D0,D1,P,ST[11:0],
2383                **              FUNCD1,RESREG
2384                **
2385                ** Stk lvls:   6 (CNTSTR)(<STOSUB>)
2386                **
2387                ** Algorithm:
2388                **      Input pending chars (CNTSTR).
2389                **      If inputting field, store stacked chars in
2390                **        variable destination.
2391                **      Return.
2392                **
2393                ** History:
2394                **    Date      Programmer              Modification
2395                **  --------    ----------    ------------------------------
2396                **  01/12/84       NZ         Updated documentation
2397                **  01/06/83       MB         Wrote routines.
2398                **
2399                ****************************************************************
2400                ****************************************************************
2401 F2840        STORFL                          Store field in expr dest.
2402 F2840 7350           GOSUB  CNTSTR           Input remaining chars.
2403 F2844 7FC9           GOSUB  AS=FTY           Get IMAGE field type to A[S]
2404 F2848 A4C            A=A-1  S                Inputting field?
2405 F284B 400            RTNC                   No. Trashing chars.
2406                *
2407 F284E 1B00           D0=(5) =STMTD1
```

```
            000
2408 F2855 146          C=DAT0 A
2409 F2858 135          D1=C                Restore D1 to the top of the stack.
2410 F285B 7699         GOSUB  CS=TYP       Get destination variable type.
2411 F285F 441          GOC    STONUM       Numeric variable goto STONUM
2412              *
2413              * IMAGE type must not be numeric, as variable is not numeric
2414              *
2415 F2862 A4C          A=A-1  S            Is the IMAGE type numeric?
2416 F2865 4A0          GOC    badtyp       Yes..."Data Type" error
2417              *
2418 F2868 7DDD         GOSUB  strhed       Generates header for string
2419 F286C 6620         GOTO   STODES       Store the string
2420              *-
2421              *-
2422 F2870 66F9 badtyp  GOTO   BADTYP       "Data Type" error
2423              *-
2424              *-
2425 F2874 844  STONUM  ST=0   MltItm       Var is numeric...check IMAGE type
2426 F2877 A4C          A=A-1  S
2427 F287A A4C          A=A-1  S            Is the IMAGE type string?
2428 F287D 42F          GOC    badtyp       Yes..."Data Type" error
2429              *
2430 F2880 7E28         GOSUB  GETNUM       Parse the number string from stack
2431 F2884 7D19         GOSUB  popstk       Pop the item off the stack into A
2432 F2888 74CD         GOSUB  D1=stk       Set D1 to AVMEME
2433 F288C 1CF          D1=D1- 16           Back up for numeric field
2434 F288F 1517 STODE1  DAT1=A W            Write out the item to the stack
2435 F2893 66D8 STODES  GOTO   STOSUB       Do the assignment to the variable
```

```
2436                    EJECT
2437              ****************************************************************
2438              ****************************************************************
2439              **
2440              ** Name:        CNTSTR - Read characters onto stack by count
2441              ** Name:        CNTST1 - Read characters by count, obey sTRASH
2442              **
2443              ** Category:    LOCAL
2444              **
2445              ** Purpose:
2446              **      Read characters onto stack, save stack pointer in STMTD1
2447              **
2448              ** Entry:
2449              **      B[A] is the number of characters to read
2450              **      STMTD1 is the current stack pointer
2451              **
2452              ** Exit:
2453              **      Carry clear
2454              **      D0 points to STMTD1
2455              **      STMTD1 contains the new stack pointer
2456              **      If an error is detected, takes a direct error exit
2457              **
2458              ** Calls:       DTOH,RESTD1,REDCHR
2459              **
2460              ** Uses.......
2461              **  Inclusive: A,B,C,D[15:13,5:0],R0-R2,D0,D1,P,STMTD1,ST[7:0]
2462              **
2463              ** Stk lvls:    5 (REDCHR)
2464              **
2465              ** Algorithm:
2466              **      CNTSTR:Set sTRASH=0  (Don't trash characters)
2467              **      CNTST1:Set sIGNOR=0  (Don't ignore any characters)
2468              **             Copy #chars from B[A] to A[A]
2469              **             Convert #chars from decimal to hex, put into A[A]
2470              **      ENTSTr:Set sCOUNT=1  (Do enter characters by count)
2471              **             Save execute pointer in R0
2472              **             If count <> 0,
2473              **               then read characters from loop (REDCHR),
2474              **                    save stack pointer in STMTD1
2475              **             Zero B[A] (count)
2476              **             Restore execute pointer
2477              **             Return.
2478              **
2479              ** History:
2480              **
2481              **    Date      Programmer              Modification
2482              **  --------    ----------    ---------------------------------
2483              **  12/19/83      NZ          Added documentation
2484              **                SC          Wrote routine
2485              **
2486              ****************************************************************
2487              ****************************************************************
2488 F2897 846  CNTSTR  ST=0   sTRASH        Don't trash.
2489 F289A 847  CNTST1  ST=0   sIGNOR        No special char to ignore.
2490 F289D D4           A=B    A             # input chars.
```

```
2491 F289F 8E00          GOSUBL =DTOH         Result in C[A]
           00
2492 F28A5 DA            A=C    A
2493 F28A7 855  ENTSTr   ST=1   sCOUNT        Count input chars.
2494 F28AA 137  ENTST2   CD1EX               Save D1 (=xqt addr) in R0.
2495 F28AD 108           R0=C
2496 F28B0 8E00          GOSUBL =RESTD1       Restore stack pointer from STMTD1.
           00
2497 F28B6 8A8           ?A=0   A            Is the input count zero?
2498 F28B9 61            GOYES  ENTST3       Yes...skip the read phase.
2499 F28BB 783A          GOSUB  REDCHR       No...input chars.
2500 F28BF 491           GOC    REDCer
2501 F28C2 1B00          D0=(5) =STMTD1      Write the stack pointer to STMTD1.
           000
2502 F28C9 137           CD1EX
2503 F28CC 144           DAT0=C A
2504 F28CF D1   ENTST3   B=0    A            Zero counter to start again.
2505 F28D1 118           C=R0                Restore D1 (=xqt addr).
2506 F28D4 135           D1=C
2507 F28D7 03            RTNCC
2508            *-
2509            *-
2510 F28D9 874  REDCer   ?ST=1  Memerr       Insufficient memory?
2511 F28DC 60            GOYES  MEMerr       Yes...go to MEMERR
2512 F28DE 6C4A          GOTO   ENTRex       No...set up the error, exit
2513            *-
2514            *-
2515 F28E2 8D00 MEMerr   GOVLNG =MEMERR       Say "Insufficient memory"
           000
2516            *-
2517            *-
2518 F28E9 8D00 D1fstk   GOVLNG =D1FSTK
           000
2519            *-
2520            *-
2521 F28F0 AC9  fndchb   C=B    S
2522 F28F3 8C00 fndchk   GOLONG =FNDCHK
           00
2523            *-
2524            *-
2525 F28F9 8C00 =getdev  GOLONG =GETDev
           00
```

```
2526                    STITLE
2527                ************************************************************
2528                ************************************************************
2529                **
2530                ** Name:        CKmode - Check if the mailbox is controller
2531                **
2532                ** Category:    PILUTL
2533                **
2534                ** Purpose:
2535                **      Check if the mailbox is the loop controller.  If it is
2536                **      not, take a direct error exit.
2537                **
2538                ** Entry:
2539                **      DO points to the selected mailbox
2540                **
2541                ** Exit:
2542                **      Carry clear
2543                **      Direct exit to error routine if not loop controller
2544                **
2545                ** Calls:       GETDev
2546                **
2547                ** Uses:        ST[3:0]
2548                **
2549                ** Stk lvls:    2 (GETDev)
2550                **
2551                ** History:
2552                **
2553                **    Date      Programmer              Modification
2554                **  --------    ----------    --------------------------------
2555                **  12/19/83       NZ         Updated documentation
2556                **                 SC         Wrote routine
2557                **
2558                ************************************************************
2559                ************************************************************
2560 F28FF 76FF =CKmode GOSUB  getdev        Check if controller
2561 F2903 500          RTNNC                Controller...return, carry clear
2562 F2906 300          LC(1)   =eBADMD      Not controller...error exit
2563 F2909 20           P=      =ePIL
2564 F290B 6F1A         GOTO    ENTRex       "Invalid Mode"
```

```
2565                    STITLE REQUEST execute
2566                ****************************************************************
2567                ****************************************************************
2568                **
2569                ** Name:        REQST - Execute the REQUEST statement
2570                **
2571                ** Category:    STEXEC
2572                **
2573                ** Purpose:
2574                **      Set up HPIL response to serial poll:
2575                **        If bit 6 of the status byte is set, this also sets
2576                **        a loop SRQ.
2577                **      If Titan is a controller at the time, it won't set
2578                **      service request but will still set up to respond to
2579                **      serial poll when later it become a device.
2580                **
2581                ** Entry:
2582                **      D0 is the PC
2583                **
2584                ** Exit:
2585                **      Through NXTSTM if no error, BSERR if error
2586                **
2587                ** Calls:       GLOOP#,GETARG,PUTE,<NXTSTM>
2588                **
2589                ** Uses.......
2590                **  Inclusive: A,B,C,D,R0-R4,D0,D1,P,STMTD0,ST[11:0],FUNCxx,
2591                **             All RAM EXPEXC is permitted to use
2592                **
2593                ** Stk lvls:    7 (GLOOP#)(GETARG)
2594                **
2595                ** History:
2596                **
2597                **    Date      Programmer              Modification
2598                **  --------    ----------    ---------------------------------
2599                **  12/20/83       NZ         Packed, changed call to GETARG to
2600                **                            call GLOOP# first to save a stack
2601                **                            level
2602                **  12/19/83       NZ         Added documentation
2603                **                 SC         Wrote routine
2604                **
2605                ****************************************************************
2606                ****************************************************************
2607 F290F 0000           REL(5) =REQSTd
           0
2608 F2914 0000           REL(5) =REQSTp
           0
2609 F2919 7000 =REQST    GOSUB  =GLOOP#        Get loop number
2610 F291D 7620           GOSUB  GETARG         Get argument
2611 F2921 3500           LC(6)  =mSETS1        Set status length=1 byte
           0000
2612 F2929 7ACB           GOSUB  pute
2613 F292D D9             C=B    A
2614 F292F F2             CSL    A
2615 F2931 F2             CSL    A
2616 F2933 3100           LC(2)  =mSETST        Load low 2 nibs of SET STATUS msg
```

```
2617 F2937 24          P=     4
2618 F2939 3100        LC(2)  =mSTS@4
2619 F293D 76BB        GOSUB  pute          Set status value to B[B] value
2620 F2941 8C54 RQSTRT GOLONG ENTRTN
           6F
2621           ****************************************************************
2622           ****************************************************************
2623           **
2624           ** Name:       GETARG - Get an argument from memory
2625           **
2626           ** Category:   LOCAL
2627           **
2628           ** Purpose:
2629           **      Get an argument which follows an (optional) loop #
2630           **      (Assumes GLOOP# has been called just before this)
2631           **
2632           ** Entry:
2633           **      All exit conditions of GLOOP#
2634           **      D0 is the PC
2635           **
2636           ** Exit:
2637           **      D0 points to the mailbox
2638           **      B[B] is the value of the argument
2639           **      Carry clear
2640           **      P=0
2641           **
2642           ** Calls:      SAVED0,FNDCHK,SWAPD0,GTYPR+,RESTD0
2643           **
2644           ** Uses.......
2645           **  Inclusive: A,B,C,D,R0-R4,D0,D1,P,STMTD0,ST[11:0],FUNCxx,
2646           **             All RAM EXPEXC is permitted to use
2647           **
2648           ** Stk lvls:   6 (GTYPR+)
2649           **
2650           ** History:
2651           **
2652           **    Date     Programmer            Modification
2653           **  --------   ----------   ------------------------------
2654           **  12/20/83      NZ        Installed fix for SR #0039-1075(1)
2655           **                          The fix involves moving the call
2656           **                          to GLOOP# to the calling routine
2657           **                          to save one RSTK level, then calling
2658           **                          GETARG
2659           **  12/19/83      NZ        Added documentation
2660           **                SC        Wrote routine
2661           **
2662           ****************************************************************
2663           ****************************************************************
2664 F2947 8E00 GETARG  GOSUBL =SAVED0       Save D0 in STMTD0 for use later
           00
2665 F294D 72AF        GOSUB  fndchk        Find the mailbox
2666 F2951 4F5         GOC    ErrorX        Error...exit
2667 F2954 8E00        GOSUBL =SWAPD0       Save mailbox addr in STMTD0, get PC
           00
2668 F295A 161         D0=D0+ 2             Skip the leading <tCOMMA>
```

```
2669 F295D 8E00          GOSUBL =GTYPR+       Get the status byte
           00
2670 F2963 4D4           GOC    ErrorX        Error...exit
2671 F2966 8C00          GOLONG =RESTDO       Restore mailbox pointer
           00
2672               *-
2673               *-
2674 F296C 7000 ENABfx   GOSUB  =GLOOP#       Get loop number
2675 F2970 73DF          GOSUB  GETARG        Get argument
2676 F2974 6E60          GOTO   ENABL1        Continue with enable code
2677               *-
2678               *-
2679 F2978 0             CON(1) =FIXSPC       3 nibbles available here
2680 F2979               BSS    3-1
2681               ****************************************************************
2682               ****************************************************************
2683               **
2684               ** Name:       CKLOP# - Read and check loop # for range
2685               ** Name:       GETLOP - Check loop # for range, put into C[S]
2686               **
2687               ** Category:   LOCAL
2688               **
2689               ** Purpose:
2690               **      Get loop number from memory, if there.  If not there,
2691               **      return loop # 1.  If there, verify that the loop # is
2692               **      in the range 1 <= l <= 3
2693               **
2694               ** Entry:
2695               **      P=0,HEXMODE
2696               **      CKLOP#:D0 points to the loop # expression, if any
2697               **      GETLOP:B[A] is the loop # (in HEX)
2698               **
2699               ** Exit:
2700               **      Carry set
2701               **      C[S] is the loop # - 1
2702               **      If an error is detected, takes a direct exit to BSERR
2703               **
2704               ** Uses:
2705               **      CKLOP#:A,B,C,D,R0-R4,D0,D1,P,FUNCxx,ST[11:0],all RAM
2706               **             EXPEXC is permitted to use
2707               **      GETLOP:A[A],C[W]
2708               **
2709               ** Stk lvls:
2710               **      CKLOP#:6 (GTYPR+)
2711               **      GETLOP:0
2712               **
2713               ** History:
2714               **
2715               **    Date     Programmer          Modification
2716               **  --------   ----------   ---------------------------------
2717               **  12/19/83      NZ        Updated documentation
2718               **  03/19/83      NZ        Modified routine
2719               **                SC        Wrote routine
2720               **
2721               ****************************************************************
```

```
2722            ****************************************************************
2723 F297B AC2  =CKLOP# C=0    S
2724 F297E 14A          A=DATO B              Read first token
2725 F2981 A80          A=0    P              (Check if it is Fx hex)
2726 F2984 A0C          A=A-1  P
2727 F2987 B64          A=A+1  B
2728 F298A 400          RTNC                  If carry, done (return C[S]=0)
2729 F298D 8E00         GOSUBL =GTYPR+        Get byte from @ PC
             00
2730 F2993 4D1          GOC    ErrorX         Error
2731 F2996 D4   =GETLOP A=B    A              Copy loop # to A[A]
2732 F2998 CC           A=A-1  A              Convert to option base zero
2733 F299A 441          GOC    outrng         If carry, too small
2734 F299D D2           C=0    A
2735 F299F 303          LC(1)  3              Set C[A]="00003"
2736 F29A2 8BE          ?A>=C  A              Is A[A] too big?
2737 F29A5 A0           GOYES  outrng         Yes...too big
2738 F29A7 A86          C=A    P              No...accept it
2739 F29AA 816          CSRC                  Put loop # into C[S]
2740 F29AD 02           RTNSC                 Set carry for exit
2741            *-
2742            *-
2743 F29AF 20   outrng  P=     =eRANGE         SAY "ARG. OUT OF RANGE"
2744 F29B1 6EA0 ErrorX  GOTO   Errorx          SAY "ARG. OUT OF RANGE"
```

```
2745                    STITLE ON INTR/ENABLE INTR execute
2746                ****************************************************************
2747                ****************************************************************
2748                **
2749                ** Name:       ONINTx - Execute the ON INTR statement
2750                **
2751                ** Category:   STEXEC
2752                **
2753                ** Purpose:
2754                **      Execute the ON INTR statement
2755                **
2756                ** Entry:
2757                **      D0 is the PC
2758                **
2759                ** Exit:
2760                **      Through NXTSTM
2761                **
2762                ** Calls:      None
2763                **
2764                ** Uses.......
2765                **  Inclusive: C[A],D0,D1
2766                **
2767                ** Stk lvls:   0
2768                **
2769                ** History:
2770                **
2771                **    Date      Programmer              Modification
2772                **  --------    ----------    --------------------------------
2773                **  12/19/83       NZ         Added documentation
2774                **                 SC         Wrote routine
2775                **
2776                ****************************************************************
2777                ****************************************************************
2778 F29B5 0000         REL(5) =ONINTd
           0
2779 F29BA 0000         REL(5) =ONINTp
           0
2780 F29BF 86D  =ONINTx ?ST=0  13              Is the machine currently running?
2781 F29C2 F0           GOYES  ENTrtn          No...don't do anything
2782 F29C4 1F00         D1=(5) =ONINTR         Yes...save the current address...
           000
2783 F29CB 136          CDOEX
2784 F29CE 145          DAT1=C A               ...in the "ONINTR" RAM location
2785 F29D1 6F6F ENTrtn  GOTO   RQSTRT          Go to NXTSTM
2786                ****************************************************************
2787                ****************************************************************
2788                **
2789                ** Name:       ENABLE - Execute the ENABLE INTR statement
2790                **
2791                ** Category:   STEXEC
2792                **
2793                ** Purpose:
2794                **      Execute the ENABLE INTR statement
2795                **
2796                ** Entry:
```

```
2797            **      D0 is the PC
2798            **
2799            ** Exit:
2800            **      Through NXTSTM if OK, BSERR if error
2801            **
2802            ** Calls:      GLOOP#,GETARG,PUTC,<NXTSTM>
2803            **
2804            ** Uses.......
2805            **  Inclusive: A,B,C,D,R0-R4,D0,D1,P,STMTD0,ST[11:0],FUNCxx,
2806            **             All RAM EXPEXC is permitted to use
2807            **
2808            ** Stk lvls:   7 (GLOOP#)(GETARG)
2809            **
2810            ** History:
2811            **
2812            **    Date      Programmer            Modification
2813            **  --------    ----------    --------------------------------
2814            **  12/21/83      NZ          Split call to GETARG into two calls;
2815            **                            one to GLOOP#, then to GETARG to
2816            **                            fix a stack level bug (see REQST)
2817            **  12/19/83      NZ          Added documentation
2818            **                SC          Wrote routine
2819            **
2820            ****************************************************************
2821            ****************************************************************
2822 F29D5 0000          REL(5) =ENABLd
           0
2823 F29DA 0000          REL(5) =ENABLp
           0
2824 F29DF 6C8F =ENABLE GOTO   ENABfx        Goto enable fix space
2825            *-
2826 F29E3 3300 ENABL1  LC(4)  =mSETIM       Set interrupt mask...
           00
2827 F29E9 AE9           C=B    B            ...to the value in B[B]
2828 F29EC 70FA          GOSUB  putc
2829 F29F0 60EF          GOTO   ENTrtn       Exit through NXTSTM
```

```
2830                    STITLE PASS CONTROL execute
2831                ****************************************************************
2832                ****************************************************************
2833                **
2834                ** Name:        PASS - Execute the PASS CONTROL statement
2835                **
2836                ** Category:    STEXEC
2837                **
2838                ** Purpose:
2839                **      Execute the PASS CONTROL statement (device specifier
2840                **      is optional)
2841                **
2842                ** Entry:
2843                **      D0 is the PC
2844                **
2845                ** Exit:
2846                **      Through NXTSTM if OK, through BSERR if error
2847                **
2848                ** Calls:       GETDID,START,CKmode,UNLPUT,TALK,PUTE,PUTGF
2849                **
2850                ** Uses.......
2851                **  Inclusive: A,B,C,D,R0-R4,D0,D1,P,STMTD1[3:0],STMTR1,ST[11:0],
2852                **             FUNCxx,All RAM EXPEXC is permitted to use
2853                **
2854                ** Stk lvls:    7 (GETDID)
2855                **
2856                ** History:
2857                **
2858                **    Date      Programmer              Modification
2859                **  --------    ----------    --------------------------------
2860                **  12/20/83       NZ         Packed 5 nibbles for future use
2861                **  12/19/83       NZ         Added documentation
2862                **                 SC         Wrote routine
2863                **
2864                ****************************************************************
2865                ****************************************************************
2866 F29F4 0000        REL(5) =PASSd
          0
2867 F29F9 0000        REL(5) =PASSp
          0
2868 F29FE 14A  =PASS  A=DAT0 B
2869 F2A01 3100        LC(2)  =tCOMMA
2870 F2A05 966         ?A#C   B              Is there a device specifier?
2871 F2A08 A0          GOYES  PASS10         Yes...process it
2872 F2A0A D3          D=0    A              No...use "LOOP"
2873 F2A0C AF2         C=0    W              *** This statement is unnecessary***
2874 F2A0F 5B0         GONC   PASS20         Go always
2875                *-
2876                *-
2877 F2A12 8E00 PASS10 GOSUBL =GETDID        Get the device specifier
          00
2878 F2A18 474         GOC    Errorx         Error
2879 F2A1B 8E00 PASS20 GOSUBL =START         Find and set up the loop
          00
2880 F2A21 4E3         GOC    Errorx         Error
```

```
2881 F2A24 77DE          GOSUB  CKmode       Make sure I'm the loop controller
2882 F2A28 96B           ?D=0   B            Is this either "LOOP" or (nothing)?
2883 F2A2B 41            GOYES  PASS30       Yes...just send TCT
2884 F2A2D 8E00          GOSUBL =UNLPUT      No...unaddress all listeners
           00
2885 F2A33 4C2           GOC    Errorx       Error
2886 F2A36 8E00          GOSUBL =TALK        Make the device the talker
           00
2887 F2A3C 432           GOC    Errorx       Error...set up code, goto BSERR
2888 F2A3F 3500 PASS30   LC(6)  =mTCT
           0000
2889 F2A47 7CAA          GOSUB  pute         Send TCT
2890 F2A4B 8E00          GOSUBL =PUTGF       Get back response from mailbox
           00
2891 F2A51 4E0           GOC    Errorx       Error
2892 F2A54 890           ?P=    =pACK        Is it an "ACKNOWLEDGE" frame?
2893 F2A57 06            GOYES  CNTR35       Yes...OK
2894 F2A59 20            P=     0
2895 F2A5B 300           LC(1)  =eNORDY      No...Device Not Ready error
2896 F2A5E 20            P=     =ePIL
2897 F2A60 6000 Errorx   GOTO   =eRRORX
2898               *-
2899               *-
2900 F2A64 0             CON(1) =FIXSPC      5 nibbles available here
2901 F2A65               BSS    5-1
```

```
2902                    STITLE CONTROL ON/OFF execute
2903                ****************************************************************
2904                ****************************************************************
2905                **
2906                ** Name:        CONTRL - Execute the CONTROL ON/OFF statements
2907                **
2908                ** Category:    STEXEC
2909                **
2910                ** Purpose:
2911                **      Execute the CONTROL ON/OFF statements (take or give up
2912                **      control on a loop)
2913                **
2914                ** Entry:
2915                **      D0 is the PC
2916                **
2917                ** Exit:
2918                **      Through NXTSTM if no error, through BSERR if error
2919                **
2920                ** Calls:       CKLOP#,FNDMBD,CHKSTS,PUTE,FNDCH-,PUTC,<NXTSTM>,
2921                **              <REST10>
2922                **
2923                ** Uses.......
2924                **  Inclusive: A,B,C,D,R0-R4,D0,D1,P,STMTD0,ST[11:0],FUNCxx,
2925                **              All RAM EXPEXC is permitted to use
2926                **
2927                ** Stk lvls:    7 (CKLOP#)
2928                **
2929                ** History:
2930                **
2931                **     Date      Programmer              Modification
2932                **   --------    ----------    ---------------------------------
2933                **   12/19/83       NZ         Added documentation
2934                **                  SC         Wrote routine
2935                **
2936                ****************************************************************
2937                ****************************************************************
2938 F2A69 0000         REL(5) =CNTRLd
           0
2939 F2A6E 0000         REL(5) =CNTRLp
           0
2940 F2A73 161  =CONTRL D0=D0+ 2              Skip the tON/tOFF token for now
2941 F2A76 710F         GOSUB  CKLOP#         Get the loop # from memory
2942 F2A7A 1F00         D1=(5) =PCADDR        (C[S] is the loop #)
           000
2943 F2A81 143          A=DAT1 A
2944 F2A84 131          D1=A                  Set D1 to the current PCADDR
2945 F2A87 177          D1=D1+ 2+6            Skip the line length, CONTROL token
2946 F2A8A 14B          A=DAT1 B              Read the tON/tOFF token
2947 F2A8D 3100         LC(2)  =tON
2948 F2A91 962          ?A=C   B              Is this CONTROL ON?
2949 F2A94 72           GOYES  CNTR40         Yes...set the controller flag
2950                *
2951                * CONTROL OFF if here
2952                *
2953 F2A96 8E00         GOSUBL =FNDMBD        Clear DISPLAY OK bits
```

```
                00
2954 F2A9C 43C          GOC     Errorx          Error
2955 F2A9F 8E00         GOSUBL  =CHKSTS         Check if reset, get status
                00
2956 F2AA5 4AB          GOC     Errorx          Error
2957 F2AA8 3500         LC(6)   =mCLRCA         Clear Controller Active state
                0000
2958 F2AB0 734A         GOSUB   pute
2959 F2AB4 4BA          GOC     Errorx
2960 F2AB7 698E CNTR35  GOTO    RQSTRT          Goto NXTSTM
2961            *-
2962            *-
2963 F2ABB AC5  CNTR40  B=C     S               Save mailbox in B[S] for REST10
2964 F2ABE 8E00         GOSUBL  =FNDCH-         Find and check the mailbox
                00
2965 F2AC4 4B9          GOC     Errorx
2966 F2AC7 3300         LC(4)   =mSETCA         Set Controller Active state
                00
2967 F2ACD 7F0A         GOSUB   putc
2968 F2AD1 4E8          GOC     Errorx
2969 F2AD4 AC9          C=B     S               Restore mailbox # from B[S]
2970 F2AD7 8C00         GOLONG  =REST10         Restore IO (readdress, etc)
                00
```

```
2971                    STITLE Zero program poll handler
2972                ********************************************************************
2973                ********************************************************************
2974                **
2975                ** Name:        hZERPG - Handler for the ZERO program poll
2976                **
2977                ** Category:    POLL
2978                **
2979                ** Purpose:
2980                **       Handle the ZERO program poll (set interrupt mask=0)
2981                **
2982                ** Entry:
2983                **       None
2984                **
2985                ** Exit:
2986                **       XM=1, carry clear
2987                **
2988                ** Calls:       SAVSTS,FNDCHK,PUTC,RESSTS
2989                **
2990                ** Uses.......
2991                **  Inclusive: A,B[S],C,DO,P,SNAPBF[37:0]
2992                **
2993                ** Stk lvls:   1 (SAVSTS) (SAVSTS saves the levels in SNAPBF)
2994                **
2995                ** History:
2996                **
2997                **    Date      Programmer              Modification
2998                **  --------    ----------    ---------------------------------
2999                **  12/19/83       NZ         Added documentation
3000                **                 SC         Wrote routine
3001                **
3002                ********************************************************************
3003                ********************************************************************
3004 F2ADD 8E00 =hZERPG GOSUBL =SAVSTS       Save 5 RSTK levels & status bits
           00
3005 F2AE3 AC1          B=0    S            Counter for which loop is next
3006 F2AE6 AC9  ZERP05  C=B    S
3007 F2AE9 760E         GOSUB  fndchk       Find that mailbox
3008 F2AED 421          GOC    ZERP10       Not found or error...exit
3009 F2AF0 3300         LC(4)  =mSETIM      Set interrupt mask to 0
           00
3010 F2AF6 76E9         GOSUB  putc
3011 F2AFA B45          B=B+1  S            Go to next loop
3012 F2AFD 58E          GONC   ZERP05       Go "always" (if fall through, done)
3013                *-
3014                *-
3015 F2B00 8E00 ZERP10  GOSUBL =RESSTS      Restore RSTK levels, D[A],ST[11:0]
           00
3016 F2B06 6F20         GOTO   RtnSXM       Return, XM=1, Carry clear
```

```
3017                    STITLE Exception poll handler
3018                ****************************************************************
3019                ****************************************************************
3020                **
3021                ** Name:        hEXCPT - Exception poll handler
3022                **
3023                ** Category:    POLL
3024                **
3025                ** Purpose:
3026                **      Handle the exception poll (check for EOL branch due)
3027                **
3028                ** Entry:
3029                **      None
3030                **
3031                ** Exit:
3032                **      If not ON INTR: XM=1, carry clear
3033                **      If ON INTR pending and due: exits through ONTIMR!
3034                **
3035                ** Calls:       FNDCHK,PUTC,<ONTIMR>
3036                **
3037                ** Uses.......
3038                **  Inclusive: A,B,C,D0,D1,P,ST[11:0] (also what ONTIMR uses)
3039                **
3040                ** Stk lvls:    3 (FNDCHK)
3041                **
3042                ** History:
3043                **
3044                **    Date      Programmer              Modification
3045                **  --------    ----------    --------------------------------
3046                **  12/19/83        NZ          Added documentation
3047                **                  SC          Wrote routine
3048                **
3049                ****************************************************************
3050                ****************************************************************
3051 F2B0A AC1   =hEXCPT B=0     S               Initialize loop counter to first
3052 F2B0D AC9   EXPT10  C=B     S
3053 F2B10 7FDD          GOSUB   fndchk          Find the current loop
3054 F2B14 412           GOC     RtnSXM          If mailbox not found, done
3055                *
3056                * FNDCHK returns with status in C[X]
3057                *
3058 F2B17 0B            CSTEX
3059 F2B19 870           ?ST=1   =sINTR          Interrupt pending?
3060 F2B1C 80            GOYES   EXPT20          Yes...see if ON INTR branch defined
3061 F2B1E B45           B=B+1   S               No...check next loop
3062 F2B21 5BE           GONC    EXPT10          Go "always" (if fall thru, OK)
3063                *-
3064                *-
3065                *
3066                * Interrupt pending on mailbox, see if ON INTR branch exists
3067                *
3068 F2B24 1F00 EXPT20  D1=(5) =ONINTR
           000
3069 F2B2B 147           C=DAT1 A
3070 F2B2E 8AE           ?C#0   A               Is the ON INTR address zero?
```

```
3071 F2B31 B0          GOYES  EXPT40        No...see if program running
3072           *
3073           * Interrupt pending, but ONINTR=0, set Except and exit for now
3074           *
3075 F2B33 850 EXPT30  ST=1   =Except
3076 F2B36 21  RtnSXM  P=     1             Clear carry and set XM
3077 F2B38 0D          P=P-1
3078 F2B3A 00          RTNSXM
3079           *-
3080           *-
3081           *
3082           * Interrupt pending and ONINTR#0, check if program running
3083           *
3084 F2B3C 86D EXPT40  ?ST=0  13            Running?
3085 F2B3F 4F          GOYES  EXPT30        No...set Except and keep waiting
3086           *
3087           * See if the ATTN key pressed
3088           *
3089 F2B41 1F00        D1=(5) =ATNFLG
           000
3090 F2B48 147         C=DAT1 A
3091 F2B4B 90E         ?C#0   P             Has the ATTN key been hit?
3092 F2B4E 5E          GOYES  EXPT30        Yes...wait for next time around
3093           *
3094           * Interrupt pending, ONINTR#0, Running; check if at end of line
3095           *
3096 F2B50 07          C=RSTK               Current PC is on third RSTK level
3097 F2B52 D5          B=C    A               save first RSTK level in B[A]
3098 F2B54 07          C=RSTK               Pop off the second RSTK level
3099 F2B56 DA          A=C    A               save it in A[A]
3100 F2B58 07          C=RSTK               Pop off the third RSTK level
3101 F2B5A 06          RSTK=C                 and push it back on
3102 F2B5C DE          ACEX   A             Get the second RSTK level from A[A]
3103 F2B5E 06          RSTK=C                 and push it back on
3104 F2B60 DD          BCEX   A             Get the first RSTK level from B[A]
3105 F2B62 06          RSTK=C                 and put it back on
3106           *
3107           * Now check if the PC is at an EOL
3108           *
3109 F2B64 131         D1=A                 Set D1 to the current PC
3110 F2B67 14B         A=DAT1 B
3111 F2B6A 3100        LC(2)  =tEOL         Check if it points to an EOL
3112 F2B6E 966         ?A#C   B             Is it at EOL?
3113 F2B71 2C          GOYES  EXPT30        No...set Except, wait for next time
3114           *
3115           * We are going to do an end-of-line branch here!!
3116           *
3117 F2B73 137         CD1EX                Save PC on stack
3118 F2B76 06          RSTK=C
3119           *
3120 F2B78 3300        LC(4)  =mSETIM       Set interrupt mask to zero
           00
3121 F2B7E 7E59        GOSUB  putc            to clear interrupt pending
3122 F2B82 1F00        D1=(5) =ONINTR
           000
```

```
3123 F2B89 147          C=DAT1 A            Read the ONINTR address again
3124 F2B8C 08           CLRST               Clear ON ERROR & ON TIMER flags
3125 F2B8E 850          ST=1   =sEXTGS      Set external flag
3126 F2B91 8D00         GOVLNG =ONTIMR      Take the jump
           000
```

```
3127                    STITLE Key definition poll handler
3128                ****************************************************************
3129                ****************************************************************
3130                **
3131                ** Name:        hKYDF - Handler for the keydef poll
3132                **
3133                ** Category:    POLL
3134                **
3135                ** Purpose:
3136                **      Handle the key def poll for HPIL key (#FF)
3137                **
3138                ** Entry:
3139                **      P=0
3140                **      R0[6:5] is the key number
3141                **
3142                ** Exit:
3143                **      If HPIL data and remote then define a colon-def key
3144                **      to execute the statement
3145                **
3146                ** Calls:       ASRC5,FNDCHK,GETHSS,D1MSTK,CHKSTK,PUTE,RDST35,
3147                **              STRPcr,D1=AVE,I/OALL
3148                **
3149                ** Uses.......
3150                **  Inclusive: A,C[B] (If not handled)
3151                **  Inclusive: A,B,C,D,R0,R1,R2,D0,D1,P (If handled)
3152                **
3153                ** Stk lvls:   4 (RDST35)    (If handled...if not, 1)
3154                **
3155                ** History:
3156                **
3157                **    Date      Programmer              Modification
3158                **  --------    ----------    --------------------------------
3159                **  01/10/84        NZ        Changed size checking to always
3160                **                            get the first 255 characters from
3161                **                            the loop, if more than 255 received
3162                **  12/21/83        NZ        Added code to force valid size
3163                **                            (<4096 nibs) for key def...check
3164                **                            is done BEFORE call to I/OALL!
3165                **  12/19/83        NZ        Added documentation
3166                **  04/01/82        SC        Wrote routine
3167                **
3168                ****************************************************************
3169                ****************************************************************
3170 F2B98 110    =hKYDF  A=R0                    Recall key number...
3171 F2B9B 8E00           GOSUBL =ASRC5           ...from A[6:5]
          00
3172 F2BA1 31FF           LCHEX  FF
3173 F2BA5 966            ?A#C   B                Is it the special HPIL key?
3174 F2BA8 E8             GOYES  RtnSXM           No...return, carry clear, XM=1
3175                *
3176                * Find out which mailbox has data available
3177                *
3178 F2BAA AC1            B=0    S                Start from mailbox #1
3179 F2BAD 7F3D DFKY10    GOSUB  fndchb           Find loop B[S] (Sets Device bit)
3180 F2BB1 4E4            GOC    NoKYDF           No mailbox has data available
```

```
3181 F2BB4 D5          B=C    A          Save status bits in B[X]
3182 F2BB6 7000        GOSUB  =GETHSS    Read mailbox's handshake bits
3183 F2BBA 454         GOC    NoKYDF     If abort, exit
3184 F2BBD 870         ?ST=1  =hsRQSR    Is this mailbox requesting service?
3185 F2BC0 80          GOYES  DFKY30     Yes...see if it has data available
3186           *
3187           * Continue on to next mailbox...this one not requesting service
3188           *
3189 F2BC2 B45 DFKY20  B=B+1  S
3190 F2BC5 57E         GONC   DFKY10     Go always
3191           *-
3192           *-
3193 F2BC8     DFKY30
3194           *
3195           * Status bits are in B[X]
3196           *
3197 F2BC8 D9          C=B    A          Recall status bits
3198 F2BCA 0B          CSTEX
3199 F2BCC 860         ?ST=0  =sDATAV    Is data available?
3200 F2BCF 3F          GOYES  DFKY20     No...try next mailbox
3201           *
3202           * Read the data from the mailbox and save it on math stack
3203           *
3204 F2BD1 7B7A        GOSUB  D1=stk     Set D1 to the top of stack
3205 F2BD5 08          CLRST
3206 F2BD7 7B49        GOSUB  CHKSTK     See if room left on stack for string
3207 F2BDB D8          B=A    A          Put memory limit into B[A]
3208 F2BDD AF2         C=0    W          Clear C[S],C[A]
3209 F2BE0 CE          C=C-1  A          Set C[A]="FFFFF"
3210 F2BE2 AFA         A=C    W          Set up flag -23 indicator, count
3211 F2BE5 8E00        GOSUBL =hCPY5s    Set up frame count opcode
          00
3212 F2BEB 7809        GOSUB  pute       Set the frame count to infinite
3213 F2BEF D9          C=B    A          Put frame count into C[A]
3214 F2BF1 2E          P=     14
3215 F2BF3 31A0        LCHEX  0A         Put <Lf> in B[15:14] (Term. char)
3216 F2BF7 AF5         B=C    W
3217 F2BFA 8EBA        GOSUBL RDST35     Read the data from the loop
          7F
3218 F2C00 454 NoKYDF  GOC    NOKYDF     Return no key def if error
3219 F2C03 713A        GOSUB  STRPcr     Strip off trailing <Cr>, if any
3220 F2C07 133         AD1EX             A[A] is address of top of stack
3221 F2C0A 8E00        GOSUBL =D1=AVE
          00
3222 F2C10 AF2         C=0    W          Clear C[5] for below
3223 F2C13 147         C=DAT1 A          C[A] is bottom of stack
3224 F2C16 E2          C=C-A  A          C[A] is string length in nibbles
3225 F2C18 81E         CSRB              Convert length to bytes (temp)
3226 F2C1B AF5         B=C    W          Put length into B[A] (for I/OALL)
3227 F2C1E AF2         C=0    W          Truncate string to 255 chars max
3228 F2C21 A6E         C=C-1  B
3229 F2C24 8BD         ?B<=C  A          Is the length currently <=255?
3230 F2C27 40          GOYES  DFKY40     Yes...leave it as is
3231 F2C29 D5          B=C    A          No...set it to 255.
3232 F2C2B C5  DFKY40  B=B+B  A          Convert back to nibbles
```

```
3233 F2C2D 07          C=RSTK              Save 1 level...I/OALL uses three
3234 F2C2F 10A         R2=C                  RSTK levels if buffer shrinks
3235 F2C32 3200        LC(3)  =bSTMXQ      Load HPIL stmt execute buffer ID
          0
3236 F2C37 8F00        GOSBVL =I/OALL      Allocate the buffer
          000
3237 F2C3E 11A         C=R2
3238 F2C41 06          RSTK=C              Restore the RSTK level from R2
3239 F2C43 470         GOC    DFKY50       Go if OK
3240              *
3241              * Not enough memory to create the stmt execute buffer
3242              *
3243              * If fail to return a key definition, set SO=0
3244              * XM is zero already
3245              *
3246 F2C46 840   NOKYDF ST=0   0            No key definition
3247 F2C49 03          RTNCC               Handled, no error
3248              *-
3249              *-
3250              *
3251              * Save the input string in the buffer
3252              * D0 points to the buffer header
3253              * D1 points to the buffer start
3254              *
3255 F2C4B 137   DFKY50 CD1EX               C[A] is the buffer start address
3256 F2C4E 162          D0=D0+ 3            D0 points to the buffer length
3257 F2C51 142          A=DAT0 A            Read the buffer length
3258 F2C54 1F00        D1=(5) =DEFADR
          000
3259 F2C5B 81C         ASRB                A[X] is the string length in bytes
3260 F2C5E 149         DAT1=A B            Write out the length to the buffer
3261 F2C61 171         D1=D1+ 2            Move to key type
3262 F2C64 BF2         CSL    W            Put buffer start address in C[5:1]
3263 F2C67 306         LC(1)  6            Type 6: colon def key
3264 F2C6A 15D5        DAT1=C 6            Write type, buffer start address
3265 F2C6E 7ED9        GOSUB  D1=stk       Set D1 to start of input string
3266 F2C72 162         D0=D0+ 3            Set D0 past the buffer header
3267              *
3268 F2C75 A6C   DFKY60 A=A-1  B            Are all characters written yet?
3269 F2C78 411         GOC    DFKY70       Yes...exit
3270 F2C7B 1C1         D1=D1- 2            No...read next character
3271 F2C7E 14F         C=DAT1 B
3272 F2C81 14C         DAT0=C B            Write the character to buffer
3273 F2C84 161         D0=D0+ 2
3274 F2C87 5DE         GONC   DFKY60       Go always
3275              *-
3276              *-
3277 F2C8A 850   DFKY70 ST=1   0            Do have a key definition
3278 F2C8D 03          RTNCC               No error
3279              *-
3280              *-
3281 F2C8F 0           CON(1) =FIXSPC      7 nibbles available here
3282 F2C90             BSS    7-1
3283 F2C96             END
```

```
 A=SLEN  Abs  992402 #F2492 -  1112   253   923
 AS=FTY  Abs  991767 #F2217 -   668  1466  1683  2403
 ASRC5   Ext                -  3171
 ATNFLG  Ext                -  3089
 AVE=D1  Ext                -   504
 Array   Abs       1 #00001 -    12   738
 BADTYP  Abs  991847 #F2267 -   740   737  2422
 BLDCON  Ext                -   419
 BytCnt  Abs       5 #00005 -    18  1337   876   910   925   927  1029  1069
 CHKASN  Ext                -   758
 CHKEOL  Ext                -   198   724
 CHKSTK  Abs  992550 #F2526 -  1283   901  3206
 CHKSTS  Ext                -  2955
 CHN#SV  Ext                -   668
 CHRCNT  Abs  993027 #F2703 -  1862  1495
=CKLOP#  Abs  993659 #F297B -  2723  2941
 CKST10  Abs  992604 #F255C -  1301  1285
=CKmode  Abs  993535 #F28FF -  2560  2881
 CLMDUT  Abs  992442 #F24BA -  1165  1036  1043  1063
=CLMODE  Abs  992462 #F24CE -  1175   906  1173
 CNTR35  Abs  993975 #F2AB7 -  2960  2893
 CNTR40  Abs  993979 #F2ABB -  2963  2949
 CNTRLd  Ext                -  2938
 CNTRLp  Ext                -  2939
 CNTST1  Abs  993434 #F289A -  2489  2003
 CNTSTR  Abs  993431 #F2897 -  2488  1933  1993  1999  2254  2402
=CONTRL  Abs  993907 #F2A73 -  2940
 COUNT   Abs  993035 #F270B -  1866  1862  2000
 COUNTC  Ext                -  1866
 CPLXER  Abs  992817 #F2631 -  1563  1552
 CS=TYP  Abs  991733 #F21F5 -   625   224   917  2324  2410
 CSLC5   Ext                -   480   482  1935
 CSRC5   Ext                -   490   492  1463  1931
 ChrTrp  Abs       7 #00007 -    22  1339   877   967
 Cmplex  Abs       3 #00003 -    14   736
 D0=PCA  Ext                -  1452
 D1=AVE  Ext                -  3221
 D1=AVS  Ext                -  1290
 D1@AVE  Ext                -  1609
 D1FSTK  Ext                -  2518
 D1MST+  Ext                -   731
 D1fstk  Abs  993513 #F28E9 -  2518   750
 D1mstk  Abs  992848 #F2650 -  1609   109   249   279   497   747   765  2340
                            -  2432  3204  3265
 DCRMNT  Ext                -  2053
 DEFADR  Ext                -  3258
 DEVADR  Abs  991682 #F21C2 -   540   177   184   242   305
 DFKY10  Abs  994221 #F2BAD -  3179  3190
 DFKY20  Abs  994242 #F2BC2 -  3189  3200
 DFKY30  Abs  994248 #F2BC8 -  3193  3185
 DFKY40  Abs  994347 #F2C2B -  3232  3230
 DFKY50  Abs  994379 #F2C4B -  3255  3239
 DFKY60  Abs  994421 #F2C75 -  3268  3274
 DFKY70  Abs  994442 #F2C8A -  3277  3269
 DTOH    Ext                -  2491
```

```
 DsLoop  Ext                  -   166
 ENABL1  Abs  993763 #F29E3 -  2826  2676
=ENABLE  Abs  993759 #F29DF -  2824
 ENABLd  Ext                  -  2822
 ENABLp  Ext                  -  2823
 ENABfx  Abs  993644 #F296C -  2674  2824
 ENDIMG  Ext                  -  1803
 ENT"/"  Abs  993189 #F27A5 -  2199  1543
 ENT"B"  Abs  993208 #F27B8 -  2252  1540
 ENT"C"  Abs  993063 #F2727 -  1927  1513
 ENT"H"  Abs  993249 #F27E1 -  2324  1519
 ENT"K"  Abs  993267 #F27F3 -  2330  1522
 ENT"P"  Abs  993065 #F2729 -  1928  1516
 ENT"R"  Abs  993069 #F272D -  1930  1546
 ENT"X"  Abs  993132 #F276C -  1999  1501
 ENT/03  Abs  993204 #F27B4 -  2206  2202
 ENT120  Abs  991171 #F1FC3 -   198   191
 ENT130  Abs  991185 #F1FD1 -   207   199   297
 ENT150  Abs  991192 #F1FD8 -   209   269
 ENT155  Abs  991205 #F1FE5 -   214   210
 ENT160  Abs  991208 #F1FE8 -   215  2339
 ENT180  Abs  991225 #F1FF9 -   224   217   220   309
 ENT190  Abs  991263 #F201F -   248   231
 ENT200  Abs  991296 #F2040 -   260   256
 ENT220  Abs  991304 #F2048 -   268   225
 ENT250  Abs  991311 #F204F -   270
 ENT300  Abs  991321 #F2059 -   273   261
 ENT302  Abs  991330 #F2062 -   278   274
 ENT305  Abs  991378 #F2092 -   297   285
 ENT310  Abs  991382 #F2096 -   300   293
 ENT320  Abs  991393 #F20A1 -   305   301
=ENTER   Abs  991064 #F1F58 -   158
 ENTERp  Ext                  -   157
 ENTFFM  Abs  993270 #F27F6 -  2331  2328
 ENTREX  Abs  991090 #F1F72 -   170   159   173   898
 ENTRTN  Abs  991112 #F1F88 -   180  2620
 ENTRex  Abs  992043 #F232B -   898   761   907   941  2203  2512  2564
 ENTST2  Abs  993450 #F28AA -  2494  2335
 ENTST3  Abs  993487 #F28CF -  2504   501  2351  2498
 ENTSTr  Abs  993447 #F28A7 -  2493  1940
 ENTU00  Abs  992641 #F2581 -  1462  1458
 ENTU05  Abs  992656 #F2590 -  1466  1549  2257
 ENTU07  Abs  992660 #F2594 -  1468  1727  2206
 ENTU09  Abs  992662 #F2596 -  1469  1672  1863
 ENTU20  Abs  992681 #F25A9 -  1478  1474
 ENTU30  Abs  992715 #F25CB -  1498  1494
=ENTUSG  Abs  992609 #F2561 -  1452
 ENTX07  Abs  993140 #F2774 -  2001  1996
 ENTa09  Abs  992858 #F265A -  1672  1560
 ENTb09  Abs  993031 #F2707 -  1863  1805  1941  2004
 ENTc09  Abs  993149 #F277D -  2004  2054  2111
 ENTdel  Abs  991099 #F1F7B -   177   208   302
 ENTdlm  Abs  992854 #F2656 -  1670  1537
 ENTend  Abs  993003 #F26EB -  1799  1534
 ENTlpb  Abs  993162 #F278A -  2107  1510
```

```
 ENTlpp  Abs  993166 #F278E -  2109  1528
 ENTlps  Abs  993164 #F278C -  2108  1525
 ENTmlt  Abs  993152 #F2780 -  2053  1507
 ENTrst  Abs  993178 #F279A -  2155  1531
 ENTrtn  Abs  993745 #F29D1 -  2785  2781  2829
 ENTstr  Abs  993119 #F275F -  1992  1504
 ENTu05  Abs  993221 #F27C5 -  2257  2352
 ENUFLD  Abs  992862 #F265E -  1675  1475
 ERROR   Ext                -   112
 ERRORX  Ext                -   170
 EXPT10  Abs  994061 #F2B0D -  3052  3062
 EXPT20  Abs  994084 #F2B24 -  3068  3060
 EXPT30  Abs  994099 #F2B33 -  3075  3085  3092  3113
 EXPT40  Abs  994108 #F2B3C -  3084  3071
 EndENT  Abs  992987 #F26DB -  1730  1704  1802
 Endfrm  Abs       3 #00003 -   15   219  1073  1078
 ErrorX  Abs  993713 #F29B1 -  2744  2666  2670  2730
 Errorx  Abs  993888 #F2A60 -  2897  2744  2878  2880  2885  2887  2891  2954
                               2956  2959  2965  2968
 Except  Ext                -  3075
 FINDA   Ext                -  1498
 FIXSPC  Ext                -   720   743  1122  2679  2900  3281
 FNDCH-  Ext                -  2964
 FNDCHK  Ext                -  2522
 FNDMBD  Ext                -  2953
 FORSTK  Ext                -   540   586
 FRAME-  Ext                -  1010
 FSTK-7  Abs  991712 #F21E0 -   586   280   881
 GETARG  Abs  993607 #F2947 -  2664  2610  2675
 GETD09  Abs  991118 #F1F8E -   183   168
 GETD10  Abs  991121 #F1F91 -   184   164
 GETDID  Ext                -   158  2877
 GETDev  Ext                -  2525
 GETHSS  Ext                -  3182
=GETLOP  Abs  993686 #F2996 -  2731
 GETN10  Abs  991435 #F20CB -   372   370
 GETN20  Abs  991455 #F20DF -   379   408
 GETN30  Abs  991471 #F20EF -   392   387
 GETN40  Abs  991476 #F20F4 -   396   380
 GETN50  Abs  991512 #F2118 -   407   405
 GETN60  Abs  991518 #F211E -   411   398   401
 GETN80  Abs  991592 #F2168 -   431   427
 GETNUM  Abs  991410 #F20B2 -   360   268  2430
 GETX    Ext                -   962
 GLOOP#  Ext                -  2609  2674
 GTYPR+  Ext                -  2669  2729
 H&Kcnt  Abs  993320 #F2828 -  2346  2342
 HDFLT   Ext                -  2263
 HorK    Abs  992902 #F2686 -  1699  1689
 I/OALL  Ext                -  3236
 IMerr   Ext                -  1563
 KorH    Abs       5 #00005 -   19   214   273   308  2338
 MEMBER  Ext                -  1493
 MEMERR  Ext                -  2515
 MEMerr  Abs  993506 #F28E2 -  2515  2511
```

```
 MFLG=0  Ext                    -   772
 MLFFLG  Ext                    -   192   753
 Memerr  Abs       4 #00004     -    17   984   997  1067  1283  1301  2510
 Mflg=0  Abs  991951 #F22CF     -   771   728  1706
 MltItm  Abs       4 #00004     -    16   215   230   307   369   386  2425
 NOKYDF  Abs  994374 #F2C46     -  3246  3218
 NRMCON  Ext                    -   420
 NUMSCN  Ext                    -   416
 NXTD10  Abs  991870 #F227E     -   747   739
 NXTD20  Abs  991939 #F22C3     -   765   756
 NXTDS-  Abs  991816 #F2248     -   730  1710
=NXTDST  Abs  991793 #F2231     -   724   207   300
 NXTEXP  Ext                    -  1708
 NXTVA-  Ext                    -   730
 NoKYDF  Abs  994304 #F2C00     -  3218  3180  3183
 NumImg  Abs  992908 #F268C     -  1701  1693
 ONINTR  Ext                    -  2782  3068  3122
 ONINTd  Ext                    -  2778
 ONINTp  Ext                    -  2779
=ONINTx  Abs  993727 #F29BF     -  2780
 ONTIMR  Ext                    -  3126
 OUTPd   Ext                    -   156
=PASS    Abs  993790 #F29FE     -  2868
 PASS10  Abs  993810 #F2A12     -  2877  2871
 PASS20  Abs  993819 #F2A1B     -  2879  2874
 PASS30  Abs  993855 #F2A3F     -  2888  2883
 PASSd   Ext                    -  2866
 PASSp   Ext                    -  2867
 PCADDR  Ext                    -  2942
 POPMTH  Ext                    -   498   748
 PUTC    Ext                    -  1179
 PUTE    Ext                    -  1192
 PUTGF   Ext                    -  2890
 RANGEN  Ext                    -   397
 RCVOFS  Ext                    -  1724
 RDATTY  Ext                    -   740
 RDS30.  Abs  992287 #F241F     -  1019  1030
 RDST01  Abs  992001 #F2301     -   883   110
 RDST05  Abs  992029 #F231D     -   894   890
 RDST10  Abs  992047 #F232F     -   901   897
 RDST15  Abs  992099 #F2363     -   927   905   909
 RDST20  Abs  992139 #F238B     -   943   913   918
 RDST25  Abs  992143 #F238F     -   946   911   928
 RDST30  Abs  992154 #F239A     -   952   947  1019
 RDST35  Abs  992171 #F23AB     -   960   953   994  3217
 RDST40  Abs  992185 #F23B9     -   964   993
 RDST45  Abs  992225 #F23E1     -   982   974   981
 RDST50  Abs  992231 #F23E7     -   984   968
 RDST52  Abs  992247 #F23F7     -   990   983
 RDST55  Abs  992252 #F23FC     -   992   966   985   998
 RDST60  Abs  992260 #F2404     -   997   987
 RDST65  Abs  992266 #F240A     -  1001   963
 RDST70  Abs  992291 #F2423     -  1022  1012
 RDST75  Abs  992301 #F242D     -  1031   961  1018
 RDST80  Abs  992307 #F2433     -  1035  1009
```

```
 RDST85  Abs  992317 #F243D -  1041  1023
 RDST87  Abs  992345 #F2459 -  1052  1047
 RDST89  Abs  992347 #F245B -  1053  1049
 RDST90  Abs  992355 #F2463 -  1063  1032
 RDST95  Abs  992397 #F248D -  1078  1070  1072
 RDST99  Abs  992400 #F2490 -  1079  1076
 RED-LF  Abs  991972 #F22E4 -   875   209
 REDCOO  Abs  991975 #F22E7 -   876   872
=REDCHR  Abs  991991 #F22F7 -   880  2499
 REDCer  Abs  993497 #F28D9 -  2510   211  2500
=REQST   Abs  993561 #F2919 -  2609
 REQSTd  Ext                -  2607
 REQSTp  Ext                -  2608
 RESSTS  Ext                -  3015
 REST10  Ext                -  2970
 RESTDO  Ext                -   283  2671
 RESTD1  Ext                -  2496
 RQSTRT  Abs  993601 #F2941 -  2620  2785  2960
 RTNCHK  Abs  991096 #F1F78 -   173  1742
 RtnSXM  Abs  994102 #F2B36 -  3076  3016  3054  3174
 S-R0-3  Ext                -   626
 S-R1-1  Ext                -  1114
 SAVEDO  Ext                -  2664
 SAVED1  Ext                -  1719
 SAVEIT  Ext                -   179   185
 SAVSTS  Ext                -  3004
 SETTRM  Abs  992524 #F250C -  1236   942
 SKP-LF  Abs  991965 #F22DD -   871  1741  2201
 START   Ext                -   760   896  2879
 STKVCT  Ext                -   732
 STMTDO  Ext                -  1714  2346
 STMTD1  Ext                -  2407  2501
 STOBIN  Abs  993225 #F27C9 -  2260  2256
 STODE1  Abs  993423 #F288F -  2434  2266
 STODES  Abs  993427 #F2893 -  2435  2419
 STONUM  Abs  993396 #F2874 -  2425  2411
 STORE   Ext                -   485
 STORFL  Abs  993344 #F2840 -  2401  1671  1676  1800  2156  2200
 STOSUB  Abs  991594 #F216A -   473   275   278  2435
 STRHED  Ext                -  1606
 STRPcr  Abs  992824 #F2638 -  1598   222  3219
 SVTRC   Ext                -   771
 SWAPDO  Ext                -  2667
 Sign    Abs       6 #00006 -    20   378   402   406   426
 StrIng  Abs  992905 #F2689 -  1700  1692
 String  Abs       2 #00002 -    13
 TALK    Ext                -  2886
=TER/LF  Abs  992509 #F24FD -  1231  1168
 TERCHR  Ext                -   878  2331
 TRESDO  Ext                -   186
 TRESD1  Ext                -   421
 TSAVD1  Ext                -   418
 Trash   Abs       6 #00006 -    21  1338   871   875   912   965  1071  1284
 TstEnd  Ext                -  1696
 Tstend  Abs  992895 #F267F -  1696  1703  1801  2341
```

```
 UNLPUT  Ext                  -  2884
=UNT     Abs  992486 #F24E6  -  1182  1174
 USGrst  Ext                  -  2157
 USING   Ext                  -   195
 USloop  Ext                  -  2109
 YTML    Ext                  -   951
 ZERP05  Abs  994022 #F2AE6  -  3006  3012
 ZERP10  Abs  994048 #F2B00  -  3015  3008
=aVE=D1  Abs  991675 #F21BB  -   504   115   296   388   473   499   749  2349
 bSTMXQ  Ext                  -  3235
 badtyp  Abs  993392 #F2870  -  2422  2416  2428
 eABORT  Ext                  -  1008  1037
 eBADMD  Ext                  -  2562
 eDSPEC  Ext                  -   169
 eNORDY  Ext                  -  2895
 ePIL    Ext                  -  1054  2563  2896
 eRANGE  Ext                  -  2743
 eRRORX  Ext                  -  2897
 eUNEXP  Ext                  -  1052
 eXPEXC  Ext                  -   729
 exit    Abs  992999 #F26E7  -  1742  1738  2343
 flEOT   Ext                  -   888
 fndchb  Abs  993520 #F28F0  -  2521  3179
 fndchk  Abs  993523 #F28F3  -  2522  2665  3007  3053
 getEOL  Abs  992995 #F26E3  -  1741   204  1737
=getdev  Abs  993529 #F28F9  -  2525   904  1166  2560
 hCPY5s  Ext                  -   955  3211
=hENTER  Abs  991028 #F1F34  -   107
=hEXCPT  Abs  994058 #F2B0A  -  3051
=hKYDF   Abs  994200 #F2B98  -  3170
=hZERPG  Abs  994013 #F2ADD  -  3004
 hsRQSR  Ext                  -  3184
 mCLRCA  Ext                  -  2957
 mSETCA  Ext                  -  2966
 mSETIM  Ext                  -  2826  3009  3120
 mSETST  Ext                  -  2616
 mSETS1  Ext                  -  2611
 mSETTC  Ext                  -  1239
 mSETTM  Ext                  -   939  1176  1178  1236
 mSFC@5  Ext                  -  1191
 mSTS@4  Ext                  -  2618
 mTCT    Ext                  -  2888
 mUNT    Ext                  -  1183
 nXTSTM  Ext                  -   180
 outrng  Abs  993711 #F29AF  -  2743  2733  2737
 pACK    Ext                  -  2892
 pENTR1  Abs  991050 #F1F4A  -   115   111
 pEOT    Ext                  -  1011
 pSTATE  Ext                  -  1046
 pTERM   Ext                  -  1022
 popstk  Abs  991653 #F21A5  -   497  2431
 putc    Abs  992480 #F24E0  -  1179   940  1177  1184  1237  1241  2828  2967
                             -  3010  3121
 pute    Abs  992503 #F24F7  -  1192   956  2612  2619  2889  2958  3212
 putefc  Abs  992498 #F24F2  -  1190  1232
```

rEV$	Ext		-	239	371				
sCOUNT	Abs	5	#00005 -	1337	2333	2493			
sDATAV	Ext		-	3199					
sEXTGS	Ext		-	3125					
sFLAG?	Ext		-	889					
sIGNOR	Abs	7	#00007 -	1339	1939	2327	2330	2489	
sINTR	Ext		-	3059					
sTRASH	Abs	6	#00006 -	1338	2002	2334	2488		
strhed	Abs	992841	#F2649 -	1606	238	260	364	2418	
tCOMMA	Ext		-	2869					
tENTER	Ext		-	1455					
tEOL	Ext		-	3111					
tON	Ext		-	2947					
tUSING	Ext		-	189					
uCPLXC	Ext		-	1551					
uDELIM	Ext		-	1536					
uHKB^	Ext		-	1472	1685				
uIMend	Ext		-	1533					
uLOOPB	Ext		-	1509					
uLOOPP	Ext		-	1527					
uLOOPS	Ext		-	1524					
uMULT	Ext		-	1506					
uRESTP	Ext		-	1530					
uSTRPT	Ext		-	1503					

Input Parameters

  Source file name is SC&ENT::MS

  Listing file name is SC/ENT::ML::-1

  Object file name is SC%ENT::MS::-1

```
                                    111111
                          0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
                        TITLE  User Utility Routines <830927.1255>
F2C96                   ABS    #F2C96        TI%HP6 address (fixed)
                *
                *       N   N  ZZZZZ   &     U   U  TTTTT  L
                *       N   N      Z  & &    U   U    T    L
                *       NN  N     Z   & &    U   U    T    L
                *       N N N    Z     &     U   U    T    L
                *       N  NN   Z     & & &  U   U    T    L
                *       N   N  Z      &  &   U   U    T    L
                *       N   N  ZZZZZ   && &   UUU     T    LLLLL
                *
                ****************************************************************
                ****************************************************************
                **
                ** Name:       SEND - Execution of the SEND command
                **
                ** Category:   STEXEC
                **
                ** Purpose:
                **      Send frame(s) on the [specified] loop
                **
                ** Entry:
                **      DO points to loop #, if any; if none, DO points to the
                **      first frame to send
                **
                ** Exit:
                **      Through NXTSTM via ENDST, or through ERRORX
                **
                ** Calls:      GLOOP#,START+,GETDev,GFTYPE,FRAMEE,GST!NO,PUTC,
                **             PUTD,PUTE,SAVEDO,SWAPDO,RESTDO,SAVE2C,REST2C,
                **             GETERR,<ENDST>
                **
                ** Uses.......
                **  Exclusive: A,B,C,D,                 DO,D1,P
                **  Inclusive: A,B,C,D,R0,R1,R2,R3,R4,DO,D1,P,ST[11:0],FUNCxx
                **
                ** Stk lvls:   7 (GLOOP#)
                **
                ** History:
                **
                **    Date     Programmer              Modification
                **   --------   ----------    --------------------------------
                **   09/26/83      NZ         Updated documentation
                **   08/02/83      NZ         Changed to not change frame count
                **                            if device mode
                **   04/01/83      NZ         Added set frame count=inf
                **   03/01/83      NZ         Updated documentation
                **
                ****************************************************************
                ****************************************************************
F2C96 0000      REL(5) =SENDd
      0
F2C9B 0000      REL(5) =SENDp
      0
F2CA0 7B41 =SEND  GOSUB  GLOOP#       Get loop number
```

```
                *
                * GLOOP# returns with the loop number in C[S]
                *
F2CA4 7000              GOSUB   =SAVEDO         Save DO in STMTDO RAM
F2CA8 D3                D=0     A               Clear D[X]
F2CAA 8E00              GOSUBL  =START+         Entry point for loop # in C[S]
      00
F2CB0 451               GOC     SENDer          Error, P=error #
                *
                * Now DO points to the mailbox, STMTDO points to input string
                *
F2CB3 7000              GOSUB   =getdev         Check if in device mode
F2CB7 4E5               GOC     SENd41          Yes...leave frame count as is
F2CBA 3500              LC(6)   (=mSETFC)+#FFFFF Set frame count to don't count
      0000
F2CC2 7000              GOSUB   =Pute
F2CC6 6770 SENDer       GOTO    SEND40          If carry set, GOTO eRRORX
                *-
                *-
F2CCA 7D51 SEND10       GOSUB   GFTYPE          Get Frame TYPE
                *
                * DO points at first character not in A-Z, A[A[S]:0] is frame
                *
F2CCE AF6               C=A     W
F2CD1 DO                A=0     A               Clear substitute value
                *
                * FRAMEE leaves DO unchanged, C[X]: frame value, B[B]: mask
                * FRAMEE also sets P=0!
                *
F2CD3 8E00              GOSUBL  =FRAMEE
      00
F2CD9 590               GONC    SEND15          If no carry, match!
                *
                * If NOT match, this is EOL!
                *
F2CDC 3200              LCHEX   FOO             Value is FOO (EOL)
      F
F2CE1 D5                B=C     A               Mask in B[B] (00)
                *
F2CE3 F2  SEND15        CSL     A
F2CE5 F2                CSL     A
F2CE7 AE9               C=B     B               Put mask in C[B], frame in C[4:2]
F2CEA 7000              GOSUB   =SAVE2C         Save in STMTR1
F2CEE 4C4               GOC     SEND5.          If carry (EOL), send it!
F2CF1 14A               A=DATO  B               Read in next token
F2CF4 3100              LC(2)   =tCOMMA         Is there an expression?
F2CF8 966               ?A#C    B
F2CFB 04                GOYES   SEND5.          No...send the frame and continue
                *
                * Now need to get the expression (One byte)
                *
F2CFD 161               DO=DO+  2               Skip the Comma token
F2D00 7871 SEND20       GOSUB   GST!NO          Get STring or Number (EXPEXC)
                *
                * GST!NO eliminates complex numbers from consideration!
```

```
105                * If number, converts to HEX and returns with # in A[A], carry
106                * clear (If overflow or <0, jumps to error)
107                * If string, returns with D1 pointing to string, D[A] = length
108                * of string (D1 needs to be decremented to next character),
109                * and carry is set.
110                * If complex, jumps to error routine
111                *
112 F2D04 7000             GOSUB  =SWAPDO       Mailbox-->DO, PC-->RAM
113 F2D08 517              GONC   SEND60        Number!
114                *
115                *
116                * String if here!
117                *
118 F2D0B 7000 SEND30  GOSUB  =REST2C       Get back the value of byte, mask
119                *
120                * C[B] is the mask, C[4:2] is the value
121                *
122 F2D0F D1             B=0    A             Clear high nibbles of B[A]
123 F2D11 AE5            B=C    B             Mask into B[B]
124 F2D14 CF             D=D-1  A             Carry if done...
125 F2D16 4A2  SENd41  GOC    SEND41        ...done!
126 F2D19 14B            A=DAT1 B             Now A[B] is the character value
127 F2D1C 1C1            D1=D1- 2             Point to next character...
128 F2D1F 0EF0           A=A&B  A             Mask the value...
129 F2D23 F6             CSR    A
130 F2D25 F6             CSR    A             Get value back into C[X]
131                *
132                * This is a hard-wired opcode calculation!!!!!!!
133                *
134 F2D27 B56            C=C+1  M             Opcode for send frame!!!
135 F2D2A 0EFA           C=C!A  A             OR in the frame value
136 F2D2E 8E00           GOSUBL =PUTC         Send the frame
          00
137 F2D34 56D            GONC   SEND30        Go if no error
138 F2D37 6000 =eRRORX GOTO   =ERRORX
139                *-
140                *-
141 F2D3B 483  SEND5.  GOC    SEND50        Go always
142                *-
143                *-
144 F2D3E 48F  SEND40  GOC    eRRORX
145 F2D41      SEND41
146                *
147                * Done with string handling
148                *
149 F2D41 7000           GOSUB  =SWAPDO       Mailbox-->RAM, PC-->DO
150                *
151                * Check if tCOMMA...if so, continue at SEND20
152                *
153 F2D45 14A            A=DAT0 B
154 F2D48 161            DO=DO+ 2
155 F2D4B 3100           LC(2)  =tCOMMA
156 F2D4F 962            ?A=C   B             Is it a comma?
157 F2D52 EA             GOYES  SEND20        Yes...more data
158 F2D54 3100           LC(2)  =tCOLON       Frame?
```

```
    159 F2D58 966          ?A#C   B            Is it a frame type?
    160 F2D5B 60           GOYES  SEND45       No...DONE!
    161 F2D5D 6C6F         GOTO   SEND10       Yes...continue
    162             *
    163             * If here, then done with processing...
    164             *
    165 F2D61 7000 SEND45  GOSUB  =RESTDO      Restore mailbox pointer
    166 F2D65 8E00         GOSUBL =GETERR      Check if detected an HPIL error
              00
    167 F2D6B 4BC  SENDEr  GOC    eRRORX       YES...report it!
    168 F2D6E 8C00         GOLONG =ENDST       Next statement after cleanup
              00
    169             *-
    170             *-
    171 F2D74 D0   SEND50  A=0    A            Fall through to send code
    172 F2D76 7000         GOSUB  =SWAPDO      (Mailbox-->D0, PC-->RAM)
    173             *
    174             * Number if here (Value in A[A])
    175             *
    176 F2D7A 7000 SEND60  GOSUB  =REST2C      Restore value of frame
    177 F2D7E D1           B=0    A
    178 F2D80 D5           B=C    A            B[A] is now the mask
    179 F2D82 F6           CSR    A
    180 F2D84 F6           CSR    A            C[X] is now the frame value
    181 F2D86 0EF0         A=A&B  A
    182 F2D8A 0E3A         C=C!A  X            Now C[X] is the frame to send
    183             *
    184             * This is a hard-wired opcode calculation!!!!!!!!!!!!!!!!
    185             *
    186 F2D8E B56          C=C+1  M            Opcode 1xxx (xxx is frame)
    187             *
    188             * Next, check if this is MTA or MLA (F04 or F02, respectively)
    189             * (OR an EOL...F00 - if so, send EOLSTR)
    190             *
    191 F2D91 B26          C=C+1  XS
    192 F2D94 A2E          C=C-1  XS           If carry, is MxA
    193 F2D97 5D0          GONC   SEND70       Not MxA...continue
    194 F2D9A 96A          ?C=0   B
    195 F2D9D 21           GOYES  SEND80       This is EOL!
    196             *
    197             * This is another hard-wired opcode calculation!!!!!!
    198             *
    199 F2D9F AA2          C=0    XS
    200 F2DA2 B56          C=C+1  M            Opcode 200x, x=4:T, x=2:L
    201 F2DA5 8E00 SEND70  GOSUBL =PUTC        Send the frame
              00
    202 F2DAB 629F         GOTO   SEND40       Check if all OK, continue
    203             *-
    204             *-
    205 F2DAF 1F00 SEND80  D1=(5) =EOLLEN
              000
    206 F2DB6 15F6         C=DAT1 7            Read in the EOL string, length
    207 F2DBA D0           A=0    A
    208 F2DBC A8A          A=C    P
    209 F2DBF 81C          ASRB                Convert to bytes
```

```
    210 F2DC2 BF6          CSR    W            C[5:0] is the string!
    211 F2DC5 AF5          B=C    W            Save in B[5:0] for SENDIT
    212 F2DC8 A0C  SEND90  A=A-1  P            Check if done
    213 F2DCB 461          GOC    SEND95       Done!
    214 F2DCE AE9          C=B    B
    215 F2DD1 BF5          BSR    W
    216 F2DD4 F5           BSR    A            Next character is ready
    217 F2DD6 8E00         GOSUBL =PUTD        Send the data byte
              00
    218 F2DDC 5BE          GONC   SEND90       Loop back if no error
    219 F2DDF 4B8          GOC    SENDEr       Go always...
    220             *-
    221             *-
    222 F2DE2 D2   SEND95  C=0    A            Clear mask, value (DATA)
    223 F2DE4 A6E          C=C-1  B            Mask is "FF", value=0
    224 F2DE7 7000         GOSUB  =SAVE2C      Save it away for next item!
    225 F2DEB 655F         GOTO   SEND41       Continue on
    226             ****************************************************************
    227             ****************************************************************
    228             **
    229             ** Name:       GLOOP# - Get loop # from RAM (if one present)
    230             **
    231             ** Category:   EXCUTL
    232             **
    233             ** Purpose:
    234             **      Get loop number from memory
    235             **
    236             ** Entry:
    237             **      D0 points to next token
    238             **
    239             ** Exit:
    240             **      P=0
    241             **      D0 points to next item on line
    242             **      C[S] is loop # [0-2]
    243             **      Carry set if no loop # given
    244             **
    245             ** Calls:      GTYPRM
    246             **
    247             ** Uses.......
    248             **  Inclusive: A,B,C,D,R0,R1,R2,R3,R4,D0,D1,P,ST[11:0],FUNCxx
    249             **
    250             ** Stk lvls:   6 (GTYPRM)
    251             **
    252             ** History:
    253             **
    254             **    Date     Programmer              Modification
    255             **  --------   ----------    ---------------------------------
    256             **  09/26/83      NZ         Updated documentation
    257             **  03/01/83      NZ         Added documentation
    258             **
    259             ****************************************************************
    260             ****************************************************************
    261 F2DEF 14A  =GLOOP# A=DAT0 B
    262 F2DF2 20           P=     0
    263 F2DF4 3100         LC(2)  =tSEMIC
```

```
264 F2DF8 AC2          C=0    S            Clear loop #...
265 F2DFB 966          ?A#C   B            Is there a loop #?
266 F2DFE 00           RTNYES              No...return
267             *
268             * Need to get the loop #
269             *
270 F2E00 161          D0=D0+ 2            Skip the leading tSEMIC
271 F2E03 8E00         GOSUBL =GTYPRM      Get type (Sequence #) from RAM
          00
272 F2E09 4D1          GOC    GLOOPE       Error
273 F2E0C 161          D0=D0+ 2            Skip the trailing tSEMIC
274             *
275             * Now B[B] is the number
276             *
277 F2E0F A6D          B=B-1  B            Decrement by 1...
278 F2E12 421          GOC    GLOOPe       Error!
279 F2E15 3120         LC(2)  2            Max loop #
280 F2E19 9E1          ?B>C   B
281 F2E1C 90           GOYES  GLOOPe       Too big!
282 F2E1E D9           C=B    A
283 F2E20 816          CSRC                Now C[S] is loop #
284 F2E23 03           RTNCC
285             *-
286             *-
287 F2E25 20   GLOOPe  P=     =eRANGE
288 F2E27 6000 GLOOPE  GOTO   =ERRORX
289             ********************************************************************
290             ********************************************************************
291             **
292             ** Name:       GFTYPE - Get frame type from RAM
293             **
294             ** Category:   EXCUTL
295             **
296             ** Purpose:
297             **      Get frame type from RAM, given string of chars
298             **
299             ** Entry:
300             **      D0 points to string of chars (<=7)
301             **
302             ** Exit:
303             **      A contains the string (A[S] is WP value)
304             **      Carry SET if error
305             **
306             ** Calls:      CONVUC,RANGEA
307             **
308             ** Uses.......
309             **  Exclusive: A[W],C[W],P,D0
310             **  Inclusive: A[W],C[W],P,D0
311             **
312             ** Stk lvls:   2 (CONVUC)
313             **
314             ** History:
315             **
316             **    Date     Programmer             Modification
317             **  --------   ----------    --------------------------------
```

```
                **  09/26/83      NZ          Updated documentation
                **  03/01/83      NZ          Added documentation
                **
                ****************************************************************
                ****************************************************************
F2E2B AF0   =GFTYPE A=0     W
F2E2E AF2           C=0     W               Could be C=0 S
F2E31 181           D0=D0-  2
F2E34 161   GFTYP1  D0=D0+  2
F2E37 14A           A=DAT0  B               Read the byte
F2E3A 8E00          GOSUBL  =CONVUC         Convert to upper case
          00
F2E40 5B0           GONC    GFTYP2          Was lower case...OK
F2E43 8E00          GOSUBL  =RANGEA         Check if in [A-Z]
          00
F2E49 4E0           GOC     GFTYP3          No...done
F2E4C 814   GFTYP2  ASRC
F2E4F 814           ASRC                    Shift around into high nibbles
F2E52 B46           C=C+1   S               Increment count of characters
F2E55 5ED           GONC    GFTYP1          Go always
                *-
                *-
F2E58 AE0   GFTYP3  A=0     B               Clear this entry!
F2E5B 94A           ?C=0    S
F2E5E 00            RTNYES                  Carry set if error
F2E60 80DF          P=C     15
                *
                * Shift A[W] left circular P*2 times
                *
F2E64 0D            P=P-1                   Set terminate on base zero count
F2E66 810   GFTYP4  ASLC
F2E69 810           ASLC
F2E6C 0D            P=P-1
F2E6E 57F           GONC    GFTYP4          Not done yet...keep shifting
                *
                * Now A[W] is zeroes, string
                *
F2E71 A46           C=C+C   S               Convert to nibbles...
F2E74 A4E           C=C-1   S               ...and to base zero...
F2E77 ACA           A=C     S               ...and copy to A[S]
F2E7A 03            RTNCC
                ****************************************************************
                ****************************************************************
                **
                ** Name:       GST!NO - Get string or number from RAM
                **
                ** Category:   EXCUTL
                **
                ** Purpose:
                **      Get string or number from RAM
                **      (If complex or out of range, exit to error)
                **
                ** Entry:
                **      D0 points to the item
                **
```

```
371            ** Exit:
372            **      Carry set: String...D1->first byte, D[A]=length(bytes)
373            **      Carry clear: Number...A[A]=Hex value
374            **
375            ** Calls:      EXPEXC,GHEXBT
376            **
377            ** Uses.......
378            **  Exclusive: A,  C,D,                    D1
379            **  Inclusive: A,B,C,D,R0,R1,R2,R3,R4,D0,D1,P,ST[11:0],FUNCxx
380            **
381            ** Stk lvls:   5 (EXPEXC)
382            **
383            ** History:
384            **
385            **    Date     Programmer             Modification
386            **  --------   ----------   ---------------------------------
387            **  09/26/83      NZ        Updated documentation
388            **  03/01/83      NZ        Added documentation
389            **
390            ********************************************************************
391            ********************************************************************
392 F2E7C 8E00 =GST!NO GOSUBL =eXPEXC       Expression execute
        00
393            *
394            * Now check if valid number or complex or NAN or ......
395            *
396            * If A[B]=#0F or 8F, than this is a string.
397            * If A[B]=(3 legal digits), than this is a number.
398            *
399 F2E82 AE6          C=A    B
400 F2E85 B06          C=C+1  P
401 F2E88 A66          C=C+C  B
402 F2E8B 96A          ?C=0   B
403 F2E8E C2           GOYES  GST!20        This is a STRING!
404 F2E90 AB6          C=A    X
405 F2E93 05           SETDEC               Check if all BCD digits...
406 F2E95 B36          C=C+1  X
407 F2E98 A3E          C=C-1  X
408 F2E9B 04           SETHEX
409 F2E9D 932          ?A=C   X
410 F2EA0 D0           GOYES  GST!10        This is a NUMBER!
411            *
412            * If here, have SOMETHING else!
413            *
414 F2EA2 20           P=     =eNNUMR       Non-numeric data
415 F2EA4 6000 GST!ER  GOTO   =ERRORX
416            *-
417            *-
418 F2EA8 20   GST!05  P=     =eRANGE
419 F2EAA 49F          GOC    GST!ER        Go always
420            *-
421            *-
422            *
423            * Number!
424            *
```

```
425 F2EAD 8E00 GST!10  GOSUBL =GHEXBT      Pop stack, Get HEX ByTe
          00
426 F2EB3 44F          GOC    GST!05       Range error
427 F2EB6 D4           A=B    A            GHEXBT returns B[A]=value
428 F2EB8 03           RTNCC               Carry clear for number
429             *-
430             *-
431             *
432             * String!
433             *
434 F2EBA BF4  GST!20  ASR    W
435 F2EBD BF4          ASR    W            Now string length in A[A]
436 F2EC0 AF2          C=0    W            (Length is in nibbles)
437 F2EC3 D6           C=A    A
438 F2EC5 81E          CSRB                Convert to bytes...
439 F2EC8 D7           D=C    A            Copy length to D[A]
440 F2ECA 17D          D1=D1+ 14           Skip string header (-2 for end)
441 F2ECD 137          CD1EX
442 F2ED0 C2           C=A+C  A            "Start" of string in C[A]...
443 F2ED2 135          D1=C                ...and in D1
444 F2ED5 02           RTNSC               Carry set for string
445 F2ED7              END
```

```
 CONVUC  Ext                  -   328
 ENDST   Ext                  -   168
 EOLLEN  Ext                  -   205
 ERRORX  Ext                  -   138   288   415
 FRAMEE  Ext                  -    81
 GETERR  Ext                  -   166
 GFTYP1  Abs  994868 #F2E34   -   326   335
 GFTYP2  Abs  994892 #F2E4C   -   332   329
 GFTYP3  Abs  994904 #F2E58   -   338   331
 GFTYP4  Abs  994918 #F2E66   -   346   349
=GFTYPE  Abs  994859 #F2E2B   -   323    71
 GHEXBT  Ext                  -   425
=GLOOP#  Abs  994799 #F2DEF   -   261    53
 GLOOPE  Abs  994855 #F2E27   -   288   272
 GLOOPe  Abs  994853 #F2E25   -   287   278   281
 GST!05  Abs  994984 #F2EA8   -   418   426
 GST!10  Abs  994989 #F2EAD   -   425   410
 GST!20  Abs  995002 #F2EBA   -   434   403
 GST!ER  Abs  994980 #F2EA4   -   415   419
=GST!NO  Abs  994940 #F2E7C   -   392   102
 GTYPRM  Ext                  -   271
 PUTC    Ext                  -   136   201
 PUTD    Ext                  -   217
 Pute    Ext                  -    67
 RANGEA  Ext                  -   330
 REST2C  Ext                  -   118   176
 RESTDO  Ext                  -   165
 SAVE2C  Ext                  -    92   224
 SAVEDO  Ext                  -    57
=SEND    Abs  994464 #F2CA0   -    53
 SEND10  Abs  994506 #F2CCA   -    71   161
 SEND15  Abs  994531 #F2CE3   -    89    82
 SEND20  Abs  994560 #F2D00   -   102   157
 SEND30  Abs  994571 #F2D0B   -   118   137
 SEND40  Abs  994622 #F2D3E   -   144    68   202
 SEND41  Abs  994625 #F2D41   -   145   125   225
 SEND45  Abs  994657 #F2D61   -   165   160
 SEND5.  Abs  994619 #F2D3B   -   141    93    97
 SEND50  Abs  994676 #F2D74   -   171   141
 SEND60  Abs  994682 #F2D7A   -   176   113
 SEND70  Abs  994725 #F2DA5   -   201   193
 SEND80  Abs  994735 #F2DAF   -   205   195
 SEND90  Abs  994760 #F2DC8   -   212   218
 SEND95  Abs  994786 #F2DE2   -   222   213
 SENDEr  Abs  994667 #F2D6B   -   167   219
 SENDd   Ext                  -    51
 SENDer  Abs  994502 #F2CC6   -    68    60
 SENDp   Ext                  -    52
 SENd41  Abs  994582 #F2D16   -   125    65
 START+  Ext                  -    59
 SWAPDO  Ext                  -   112   149   172
 eNNUMR  Ext                  -   414
 eRANGE  Ext                  -   287   418
=eRRORX  Abs  994615 #F2D37   -   138   144   167
 eXPEXC  Ext                  -   392
```

```
getdev  Ext              -    64
mSETFC  Ext              -    66
tCOLON  Ext              -   158
tCOMMA  Ext              -    95   155
tSEMIC  Ext              -   263
```

Input Parameters

  Source file name is NZ&UTL::MS

  Listing file name is NZ/UTL:TI:ML::-1

  Object file name is NZ%UTL:TI:MS::-1

```
                                  111111
                        0123456789012345
```
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *
 2            *
 3            *       N   N  ZZZZZ   &     BBBB   III  FFFFF
 4            *       N   N      Z  & &    B   B   I   F
 5            *       NN  N     Z   & &    B   B   I   F
 6            *       N N N    Z     &     BBBB    I   FFFF
 7            *       N  NN   Z     & & &  B   B   I   F
 8            *       N   N  Z      &  &   B   B   I   F
 9            *       N   N  ZZZZZ   && &  BBBB  III  F
10            *
11                    TITLE  Basic interface <840124.1345>
12 F2ED7              ABS    #F2ED7        TI%HP6 address (fixed)
```

```
                        STITLE Cold start handler
                ****************************************************************
                ****************************************************************
                **
                ** Name:        PILCST - HPIL cold start handler routine
                **
                ** Category:    POLL
                **
                ** Purpose:
                **      Diamond cold start POLL handler routine
                **
                ** Entry:
                **      P=0, HEXMODE
                **
                ** Exit:
                **      Carry clear, XM=1, P=0
                **
                ** Calls:       I/OALL,FNDMBX,GETERR,CHKST+,D1=DSP,D1=DST
                **
                ** Uses.......
                **  Exclusive:      B[W],C[W],        R0,   D1,P
                **  Inclusive: A[W],B[W],C[W],D[15:5],R0,D0,D1,P
                **
                ** Stk lvls:    2 (FNDMBX)(I/OALL)(CHKST+)(GETERR)
                **
                ** Detail:
                **      Reset all HPIL mailboxes, set up LOOPST and DSPSET,
                **      set DISPLAY IS DISPLAY, PRINTER IS PRINTER
                **
                ** History:
                **
                **    Date      Programmer            Modification
                **   --------   ----------    --------------------------------
                **  07/26/83       NZ         Added check for Diamond error
                **                            after resetting it
                **  06/30/83       NZ         Added wakeup of Diamond after
                **                            RESET (to be sure Manual Mode bit
                **                            is clear)
                **  03/15/83       NZ         Removed check for RAM changed
                **  02/22/83       NZ         Changed CLEAR of mailboxes into
                **                            RESET of mailboxes
                **  02/11/83       NZ         Added save of D[A] in R0
                **  12/21/82       NZ         Updated documentation
                **
                ****************************************************************
                ****************************************************************
F2ED7           =PILCST
                *
                * PIL buffer (used by PILCNF to determine if HPIL was present
                * at the last configuration before current one - if not, then
                * calls PILCST as a subroutine)
                *
F2ED7           PILCS0
F2ED7 D1                B=0     A               Allocate 0 nibs (no info to store)
F2ED9 DB                C=D     A
```

```
 68 F2EDB 108          RO=C                Save D[A] in RO (I/OALL uses D[A])
 69 F2EDE 3200         LC(3)  =bPILSV
          0
 70 F2EE3 8F00         GOSBVL =I/OALL      I/O ALLocate
          000
 71 F2EEA 118          C=RO
 72 F2EED D7           D=C    A            Restore D[A] from RO
 73            *
 74            * Now reset all HP-IL mailboxes (Up to 16 of them!)
 75            *
 76 F2EEF AF2          C=0    W
 77 F2EF2 27           P=     7
 78 F2EF4 308          LC(1)  8            Reset the mailbox
 79 F2EF7 AF5          B=C    W            Save the message in B[8:0]
 80 F2EFA AC9  PILCS3  C=B    S            Find out which mailbox I'M on...
 81 F2EFD 8E00         GOSUBL =FNDMBX      ...and see if it's there
          00
 82 F2F03 491          GOC    PILCS4       Not there...no more mailboxes
 83 F2F06 AF9          C=B    W            Found one...reset it
 84 F2F09 15C8         DAT0=C 9            Reset the mailbox, clear NRD bit
 85 F2F0D 8E00         GOSUBL =GETERR      Wake it up, read the error message
          00
 86            *                           (ignore any error message here)
 87 F2F13 7ED2         GOSUB  CHKST+       Set up parameters
 88 F2F17 B45          B=B+1  S            Increment to next mailbox
 89 F2F1A 5FD          GONC   PILCS3       Go always (carry= >16 mailboxes)
 90            *-
 91            *-
 92 F2F1D      PILCS4
 93            *
 94            * Now initialize the IS-TBL
 95            *
 96 F2F1D 7063         GOSUB  D1=DSP
 97            *
 98            * Set IS-DSP ="03F1FFF", IS-PRT="02F1FFF", IS-INP="FFFFFFF",
 99            * IS-PLT="FFFFFFF"
100            *
101 F2F21 20           P=     0            FNDMBX leaves P#0 when not found
102 F2F23 36FF         LCHEX  03F1FFF
          F1F3
          0
103 F2F2C 15D6         DAT1=C 7            Write IS-DSP entry
104 F2F30 176          D1=D1+ 7
105 F2F33 36FF         LCHEX  02F1FFF
          F1F2
          0
106 F2F3C 15D6         DAT1=C 7            Write IS-PRT entry
107           *
108           * Now enable the loop (LoopOK bit of DSPSET)
109           *
110 F2F40 D2           C=0    A
111 F2F42 1D00         D1=(2) =LOOPST
112 F2F46 15D0         DAT1=C 1            Clear Offed, InptOK
113 F2F4A 7C33         GOSUB  D1=DST       Clear DispOK, set LoopOK
114           *
```

```
115             * Set LoopOK until proven wrong
116             * Set Display to restart and check device ID
117             *
118 F2F4E 307           LC(1)  7            *DispOK, Printr, Wallby, LoopOK
119 F2F51 15D0          DAT1=C 1            Write bits out to RAM
120             *
121             * Set terminating character to LF for ENTER
122             *
123 F2F55 1E00          D1=(4) =TERCHR
          00
124 F2F5B 31A0          LCHEX  0A
125 F2F5F 14D           DAT1=C B
126             *
127             * Done
128             *
129 F2F62 21    =RTNCCX P=     1
130 F2F64 0D            P=P-1               Clear the carry...
131 F2F66 00            RTNSXM              ...and set XM
```

```
                        STITLE No key wakeup poll handler
                    ******************************************************************
                    ******************************************************************
                    **
                    ** Name:       PILWNK - Wakeup, no key poll handler
                    **
                    ** Category:   POLL
                    **
                    ** Purpose:
                    **      Deep sleep wakeup-no key
                    **
                    ** Entry:
                    **      None
                    **
                    ** Exit:
                    **      Carry clear, XM=1, P=0
                    **
                    ** Calls:      None
                    **
                    ** Uses.......
                    **  Inclusive: C[P],D1
                    **
                    ** Stk lvls:   0
                    **
                    ** NOTE: Must not alter D[A] or STATUS
                    **
                    ** History:
                    **
                    **    Date     Programmer              Modification
                    **  --------   ----------    ------------------------------
                    **  12/21/82      NZ         Updated documentation
                    **
                    ******************************************************************
                    ******************************************************************
F2F68 80E   =PILWNK SREQ?               First check if SRQ pending
F2F6B 834           ?SR=0
F2F6E F1            GOYES  PILWNx       Not me (no SRQ)
                    *
                    * Check if this is a Diamond service request...if so, wake up
                    * the HP-71 by simulating the ATTN key (Setting ATNFLG#0)
                    * (Should really set ATNFLG = "F" to say "ATTN pressed once")
                    *
F2F70 824           SR=0
F2F73 0B            CSTEX
F2F75 860           ?ST=0  =sDIAsr
F2F78 20            GOYES  WNK00
F2F7A 0B     WNK00  CSTEX
F2F7C 401           GOC    PILWNx
F2F7F 1F00          D1=(5) =ATNFLG
      000
F2F86 301           LC(1)  1
F2F89 1550          DAT1=C P
                    *
                    * Now exit, carry clear, XM set
                    *
```

```
    186 F2F8D 64DF PILWNx  GOTO   RTNCCX        Return, clear carry, set XM
```

```
                        STITLE Configuration handler
                ****************************************************************
                ****************************************************************
                **
                ** Name:        PILCNF - Configuration poll handler for HPIL
                ** Name:        PILWKP - Deep-sleep wakeup poll (no processing)
                **
                ** Category:    POLL
                **
                ** Purpose:
                **      Configuration entry point - Restore buffers, set DSPCHX
                **      to address of display routine, etc
                **
                ** Entry:
                **      P=0,HEXMODE
                **
                ** Exit:
                **      Carry clear, XM=1, P=0
                **
                ** Calls:       RESTOR,I/ORES,PILCST,D1=DST,D1=DSP,D1=DSX,
                **              CHKASN,(PILWKs)
                **
                ** Uses.......
                **  Exclusive:      B[A],C[W],              D0,D1,P
                **  Inclusive: A[W],B[W],C[W],D[15:5],R0,D0,D1,P
                **
                ** NOTE: Must NOT alter D[A], Status
                **
                ** Stk lvls:    3 (PILCST)(CHKASN)
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------    ---------------------------------
                **  02/25/83       NZ         Moved IS-DSP check and DSPCHX set
                **                            later in the code
                **  02/18/83       NZ         Added check for IS-DSP before
                **                            setting DSPCHX
                **  02/11/83       NZ         Updated documentation (uses D,R0)
                **  12/21/82       NZ         Updated documentation
                **
                ****************************************************************
                ****************************************************************
F2F91           =PILCNF
F2F91 3200 0            LC(3)  =bPILSV      Check if save buffer is here
F2F96 7190              GOSUB  I/ores       Restore it
F2F9A 460               GOC    PILCN1       Found it...continue
                *
                * Save buffer not found...therefore HPIL was not present at
                * last configuration poll...need to reset Diamond, set it up
                *
F2F9D 763F              GOSUB  PILCST       Go through my coldstart code
                *
F2FA1           PILCN1
```

```
241                *
242                * Set the display device to be restarted with next character
243                *
244 F2FA1 75E2             GOSUB  D1=DST
245 F2FA5 1572             C=DAT1 XS           Read display status...
246 F2FA9 0B               CSTEX
247 F2FAB 840              ST=0   =DispOK      Set the display to be restarted
248 F2FAE 0B               CSTEX
249 F2FB0 1552             DAT1=C XS           ...Write it back out
250                *
251                * Clear the OFFed bit in each device
252                *
253 F2FB4 8E00             GOSUBL =RESTOR
          00
254                *
255                * Now reclaim all I/O buffers I use
256                * Reclaim IS-DSP, IS-PRT, bSTMXQ (shouldn't be needed), bPILAI
257                *
258 F2FBA 1B00             D0=(5) (=IS-DSP)+3  Check if I/O buffer type
          000
259 F2FC1 7850             GOSUB  PILWKs       Restore IS-DSP if needed
260 F2FC5 166              D0=D0+ 7            Next entry
261 F2FC8 7150             GOSUB  PILWKs       Restore IS-PRT if needed
262 F2FCC 20               P=     0            (PILWKs leaves P#0)
263                *
264 F2FCE 3200             LC(3)  =bSTMXQ
          0
265 F2FD3 7450             GOSUB  I/ores       Restore HPIL stmt execute buffer
266                *
267 F2FD7 3200             LC(3)  =bPILAI
          0
268 F2FDC 7B40             GOSUB  I/ores       Restore the ASSIGNIO buffer
269 F2FE0 7D92             GOSUB  D1=DSP       Check if a display is assigned
270 F2FE4 15F6             C=DAT1 7            Read it in
271 F2FE8 8E00             GOSUBL =CHKASN      Check if assigned
          00
272 F2FEE 4A2              GOC    RTNCCx       Not assigned...leave DSPCHX alone
273 F2FF1 7E92             GOSUB  D1=DSX       Display location...
274 F2FF5 147              C=DAT1 A            Read it first...
275 F2FF8 8AE              ?C#0   A
276 F2FFB E1               GOYES  RTNCCx       Exit if occupied
277 F2FFD 7500             GOSUB  PILxxx       Get address of REL(5) on RSTK...
278                *-
279 F3001 0000             REL(5) =BDISPJ      Offset to display entry
          0
280                *-
281                *-
282 F3006 07   PILxxx      C=RSTK              ...pop it off...
283 F3008 D5               B=C    A            ...save address in B[A]...
284 F300A 135              D1=C                ...and set D0 to offset
285 F300D 147              C=DAT1 A            Read in display offset...
286 F3010 C9               C=B+C  A            ...to get address of display jump
287 F3012 7D72             GOSUB  D1=DSX       Point back to entry
288 F3016 145              DAT1=C A            Write out display routine address
289                *
```

```
290 F3019       =PILWKP
291 F3019 684F  RTNCCx  GOTO   RTNCCX
292             *-
293             *-
294 F301D 146   PILWKs  C=DATO A            Read in ID, type
295 F3020 80D0         P=C    0            P=type
296 F3024 884          ?P#    4            Single I/O buffer?
297 F3027 00           RTNYES              No...return (No buffer)
298             *
299             * I/O buffer...restore it
300             *
301 F3029 F6           CSR    A            ID in C[X] now
302 F302B 8D00 I/ores  GOVLNG =I/ORES       Restore the I/O buffer
          000
```

```
                        STITLE Power-off poll handler
                ******************************************************************
                ******************************************************************
                **
                ** Name:        PILPOF -  Handler for power-off poll
                **
                ** Category:    POLL
                **
                ** Purpose:
                **      Power-off code for HPIL:
                **      -Sets device codes (DISPLAY, PRINTER, KEYBD, PLOTTER)
                **        to power-off values (to allow restart on next usage)
                **      -If flPDWN is clear and the OFFED flag is clear, sends
                **        power-down message to all Diamonds (up to 16) which
                **        are not in manual mode and are controller
                **
                ** Entry:
                **      P=0,HEXMODE
                **
                ** Exit:
                **      Carry clear, XM=1
                **
                ** Calls:       RESTRT,SFLAG?,FNDMBX,CHKSTS,PUTC+
                **
                ** Uses.......
                **  Exclusive:      B[S],C[W],D0,P,ST[11:0]
                **  Inclusive: A[W],B[W],C[W],D0,P,ST[11:0]
                **
                ** Stk lvls:    3 (RESTRT)(CHKSTS)
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------    ---------------------------------
                **  03/29/83       NZ         Added check of flPDWN flag before
                **                            powering down the loops
                **  12/21/82       NZ         Updated documentation
                **
                ******************************************************************
                ******************************************************************
F3032 7750      =PILPOF GOSUB  RESTRT        Restart all devices on loop
                *
                * Check if loop is OFFED (by OFFIO)
                *
F3036 1A00              D0=(4) =LOOPST
      00
F303C 1562              C=DAT0 XS
                *
                * =Offed is 11
                *
F3040 A26               C=C+C  XS            If carry, OFFED
F3043 454               GOC    PILP03        If carry (=Offed), exit
                *
                * Check if powerdown inhibit flag is set
                *
```

```
357 F3046 DB          C=D    A            Save D[A] in D0 (SFLAG? puts D0
358 F3048 134         D0=C                into D[A] to save D0)
359 F304B 3100        LC(2)  =flPDWN      Check if power down inhibited
360 F304F 8E00        GOSUBL =sFLAG?
          00
361 F3055 433         GOC    PILP03       If carry, just return
362             *
363             * Now shut down all the loops...
364             *
365 F3058 AC1         B=0    S            Initialize loop counter
366 F305B AC9 PILP01  C=B    S
367 F305E 8E00        GOSUBL =FNDMBX
          00
368 F3064 442         GOC    PILP03       No more mailboxes
369 F3067 8E00        GOSUBL =CHKSTS      Check status, RESET
          00
370 F306D 451         GOC    PILP02       In manual mode...leave it alone
371             *
372             * C[X] is the device status from Diamond
373             *
374 F3070 0A          ST=C
375 F3072 860         ?ST=0  =sCONTR      Am I controller?
376 F3075 E0          GOYES  PILP02       No...try next loop
377             *
378             * OK to power down this loop
379             *
380 F3077 20          P=     0
381 F3079 3100        LC(2)  =mPDLOP      Power down loop
382 F307D 8E00        GOSUBL =PUTC+       Send it
          00
383             *
384             * Don't check carry...even if carry set, continue with the
385             * other loops (if any)
386             *
387 F3083 B45 PILP02  B=B+1  S            Increment loop counter
388 F3086 54D         GONC   PILP01       Go always (if carry, > 16 loops)
389             *
390             * Done with power-off processing
391             *
392 F3089 68DE PILP03  GOTO   RTNCCX       Return, carry clear, XM set
```

```
                        STITLE Restart HPIL to search
                    ****************************************************************
                    ****************************************************************
                    **
                    ** Name:       RESTRT - Restart all HPIL devices (readdress)
                    **
                    ** Category:   PILUTL
                    **
                    ** Purpose:
                    **      Restart all device addresses in the HPIL system
                    **      (set to search for address at next access)
                    **
                    ** Entry:
                    **      P=0, HEXMODE
                    **
                    ** Exit:
                    **      P=0
                    **      Carry clear
                    **
                    ** Calls:      RESTRs,CSRC5,CSLC5,FIBOFF
                    **
                    ** Uses.......
                    **  Exclusive:      C[W],D0,P
                    **  Inclusive: A[W],C[W],D0,P
                    **
                    ** Stk lvls:   2 (FIBOFF)
                    **
                    ** History:
                    **
                    **    Date     Programmer              Modification
                    **  --------   ----------    ---------------------------------
                    **  06/01/83      NZ         Added call to FIBOFF
                    **  12/21/82      NZ         Updated documentation
                    **
                    ****************************************************************
                    ****************************************************************
F308D               =RESTRT
F308D 137                   CD1EX
F3090 8E00                  GOSUBL =CSLC5       Save D1 in C[9:5]
      00
F3096 8F00                  GOSBVL =FIBOFF      Restart FIB buffers
      000
F309D 8E00                  GOSUBL =CSRC5       Recall D1 to C[A]
      00
F30A3 135                   D1=C                Restore D1
F30A6 1B00                  D0=(5) =DSPSET
      000
F30AD 307                   LC(1)  7            DispOK=0; Wallby,Printr,LoopOK=1
F30B0 15C0                  DAT0=C 1            Write them out
                    *
                    * Now deassign all devices
                    *
F30B4 1A00                  D0=(4) =IS-DSP      Point to IS-DSP, set it OFF
      00
F30BA 7800                  GOSUB  RESTRs       IS-DSP
```

```
443 F30BE 7400          GOSUB  RESTRs        IS-PRT
444 F30C2 7000          GOSUB  RESTRs        IS-INP
445               *
446               * Fall into RESTRs for IS-PLT (exit when done with RESTRs)
447               *
448 F30C6       RESTRs
449               *
450               * D0 points to the entry
451               *
452 F30C6 15E6          C=DAT0 7
453 F30CA 23            P=     3              Check if C[3]="F"...if so, not HP
454 F30CC B06           C=C+1  P              If C[3]="F", then not HPIL/done
455 F30CF 401           GOC    RESTs4         Not HPIL or assigned to *
456 F30D2 B26           C=C+1  XS             If C[XS]="F", leave this alone
457 F30D5 4A0           GOC    RESTs4         Increment D0, return
458 F30D8 D2            C=0    A
459 F30DA CE            C=C-1  A
460 F30DC 15C2          DAT0=C 3              Write out "FFF"
461 F30E0 166  RESTs4   D0=D0+ 7              Move to the next entry
462 F30E3 20            P=     0
463 F30E5 03            RTNCC
```

```
464                     STITLE Main loop poll handler
465                 ****************************************************************
466                 ****************************************************************
467                 **
468                 ** Name:        PILMLP - HPIL handler for main loop
469                 **
470                 ** Category:    POLL
471                 **
472                 ** Purpose:
473                 **      Main loop handler code - if display is not offed,
474                 **      set ST[LoopOK] true
475                 **
476                 ** Entry:
477                 **      P=0,HEXMODE
478                 **
479                 ** Exit:
480                 **      Carry clear,XM=1
481                 **
482                 ** Calls:       D1=DST
483                 **
484                 ** Uses.......
485                 **  Inclusive: C[XS],D1,P
486                 **
487                 ** Stk lvls:    1 (D1=DST)
488                 **
489                 ** History:
490                 **
491                 **    Date      Programmer              Modification
492                 **  --------    ----------    ---------------------------------
493                 **  12/21/82       NZ         Updated documentation
494                 **  01/17/83       NZ         Changed Search from 4 to 5 (START
495                 **                            is now using ST[4] also)
496                 **
497                 ****************************************************************
498                 ****************************************************************
499 F30E7 1F00 =PILMLP D1=(5) =LOOPST        First check if loop is "OFFED"
          000
500 F30EE 1572          C=DAT1 XS
501 F30F2 0B            CSTEX
502 F30F4 870           ?ST=1  =Offed        Is it offed?
503 F30F7 20            GOYES  PILM05        Set carry if yes
504 F30F9 0B    PILM05  CSTEX
505 F30FB 451           GOC    PILMRC        If offed, just return
506                 *
507                 * Not OFFED by OFFIO...set loop OK true here
508                 *
509 F30FE 7881          GOSUB  D1=DST
510 F3102 1572          C=DAT1 XS
511 F3106 0B            CSTEX
512 F3108 850           ST=1   =LoopOK       Set Loop OK flag true again
513 F310B 0B            CSTEX
514 F310D 1552          DAT1=C XS            Write out the statuses
515 F3111 605E PILMRC   GOTO   RTNCCX        Return w/carry clear, XM=1
```

```
                        STITLE Service Request Handler
                    ****************************************************************
                    ****************************************************************
                    **
                    ** Name:        PILSRQ - HPIL service request handler
                    **
                    ** Category:    POLL
                    **
                    ** Purpose:
                    **      HPIL service request poll handler - determine SRQ
                    **      source, process SRQ
                    **
                    ** Entry:
                    **      P=0,HEXMODE
                    **
                    ** Exit:
                    **      Carry clear,P=0,XM=1
                    **
                    ** Calls:       SAVSTS,FNDMBX,GETHSS,CHKSET,PUTCN,GETST-,SFLAG?,
                    **              RESSTS
                    **
                    ** Uses.......
                    **  Exclusive:      B[A],C[W],              D1,P
                    **  Inclusive: A[W],B[A],C[W],D[15,5],D0,D1,P,SNAPBF[37:0]
                    **
                    ** Stk lvls:    0 (SAVSTS,RESSTS save them)
                    **
                    ** NOTE: Must NOT use RSTK levels OR status bits
                    **
                    ** Algorithm:
                    **      Check if Diamond SRQ...if not, return
                    **      Find which Diamond is requesting service
                    **      Check if interrupt pending...if pending, set exception
                    **      Check if data available and remote mode and "dormant":
                    **        if so, set up HPIL external key
                    **      If not interrupt and not (data available and remote)
                    **        then continue checking with next loop
                    **
                    ** History:
                    **
                    **    Date      Programmer              Modification
                    **  --------    ----------    --------------------------------
                    **  10/20/83       NZ         Implemented ER #39-10744 (if the
                    **                            first loop requesting service
                    **                            does not have anything to do, try
                    **                            any other loops for SRQ)
                    **  12/21/82       NZ         Updated documentation
                    **
                    ****************************************************************
                    ****************************************************************
F3115 65B0 PILS07   GOTO   PILS9+          Out of range for GOC
                    *-
                    *-
F3119 80E  =PILSRQ  SREQ?                  First check this is HPIL
F311C 834           ?SR=0
```

```
571 F311F 6F           GOYES  PILS07       No request pending...exit
572 F3121 824          SR=0
573 F3124 0B           CSTEX
574 F3126 860          ?ST=0  =sDIAsr      Diamond SRQ?
575 F3129 20           GOYES  PILS00       Set carry if not HPIL
576 F312B 0B   PILS00  CSTEX
577 F312D 47E          GOC    PILS07       Not HPIL...exit
578            *
579            * This is an HPIL SRQ...service it
580            *
581 F3130 07           C=RSTK               Save calling level
582 F3132 7423         GOSUB  SAVSTS       Save status, 6 levels, D[A]
583 F3136 1F00         D1=(5) =MBOX^
           000
584 F313D 147          C=DAT1 A            Save old MBOX^ value in B[3:1]
585 F3140 F2           CSL    A
586 F3142 D5           B=C    A            Mbox value in B[3:1], # in B[0]
587            *                           Set up for mbox #1
588 F3144 816  PILS20  CSRC                 Shift mailbox number into C[S]
589 F3147 8E00         GOSUBL =FNDMBX      Look for the mailbox
           00
590 F314D 486          GOC    PILS50       Not found...done
591 F3150 7B70         GOSUB  GETHSS       Read handshake nibbles (2)
592 F3154 870          ?ST=1  =hsRQSR      Requesting service?
593 F3157 90           GOYES  REQSER       Yes...see what it is
594 F3159 E5   PILS23  B=B+1  A            No...try next mailbox
595 F315B D9           C=B    A
596 F315D 56E          GONC   PILS20       Go always (if more than 16, no)
597            *-
598            *-
599            *
600            * Diamond requesting service pointed to by D0
601            *
602 F3160 7A70 REQSER  GOSUB  CHKSET       Check if this loop was reset
603 F3164 3300         LC(4)  =mSTSTC      Request status & clear SRQ
           00
604 F316A 8E00         GOSUBL =PUTCN
           00
605 F3170 8E00         GOSUBL =GETST-      Read the mailbox's status
           00
606 F3176 5B0          GONC   REQS10       (OK)
607 F3179 890          ?P=    =eABORT      Error from ATTN key hit?
608 F317C A3           GOYES  PILS50       Yes...exit routine NOW
609 F317E F6           CSR    A            No...status is in C[3:1]
610 F3180 20           P=     0            (P was =ePIL)
611 F3182 0B   REQS10  CSTEX
612            *
613            * Check if there is an interrupt pending
614            *
615 F3184 860          ?ST=0  =sINTR       Interrupt pending?
616 F3187 80           GOYES  REQS30       No...check if data is available
617 F3189 850          ST=1   =Except      Yes...set exception flag and exit
618 F318C 5CC          GONC   PILS23       Go always...check next for remote ke
619            *-
620            *-
```

```
621 F318F       REQS30
622             *
623             * Check if there is data available
624             *
625 F318F 860           ?ST=0  =sDATAV       Data available?
626 F3192 7C            GOYES  PILS23        No...try next mailbox
627             *
628             * Data is available...check if Diamond is in remote mode
629             *
630 F3194 860           ?ST=0  =sRMOTE       Remote mode?
631 F3197 2C            GOYES  PILS23        No...ignore the data, try next mbox
632             *
633             * Data available, remote mode...check if the HP-71 is dormant
634             *
635 F3199 3100          LC(2)  =flDORM
636 F319D 8E00          GOSUBL =sFLAG?       Check the dormant flag
          00
637 F31A3 55B           GONC   PILS23        Not dormant...try next mailbox
638             *
639             * Data available, remote mode, dormant...generate special key
640             *
641 F31A6 1F00          D1=(5) =KEYPTR
          000
642 F31AD 321F          LCHEX  FF1
          F
643 F31B2 1553          DAT1=C X             Set to one key, keycode = "FF"
644             *
645             * Restore MBOX^ value, restore status, RSTK, D[A], and exit
646             *
647 F31B6 D9    PILS50  C=B    A
648 F31B8 F6            CSR    A             Get mailbox # back to C[X]
649 F31BA 1F00          D1=(5) =MBOX^
          000
650 F31C1 15D2          DAT1=C 3             Restore the mailbox address
651 F31C5 75C2          GOSUB  RESSTS        Restore status, 6 levels, D[A]
652 F31C9 06            RSTK=C               Restore last stack level
653 F31CB 669D  PILS9+  GOTO   RTNCCX        Clear carry, set XM
654             ********************************************************************
655             ********************************************************************
656             **
657             ** Name:       GETHSS - Get 2 handshake nibbles from Diamond
658             **
659             ** Category:   PILI/O
660             **
661             ** Purpose:
662             **      Read the two handshake nibbles from Diamond to the HP-71
663             **      and put into ST[7:0]
664             **
665             ** Entry:
666             **      D0 points to HPIL mailbox
667             **
668             ** Exit:
669             **      The two handshake nibbles from Diamond are in ST[7:0]
670             **      Carry clear
671             **
```

```
672                ** Calls:      None
673                **
674                ** Uses:
675                **  Inclusive: ST[7:0]
676                **
677                ** Stk lvls:   0
678                **
679                ** History:
680                **
681                **    Date     Programmer          Modification
682                **  --------   ----------   --------------------------------
683                **  09/29/83      NZ        Updated documentation
684                **  04/01/83      SC        Wrote routine
685                **
686                ******************************************************************
687                ******************************************************************
688 F31CF 0B    =GETHSS CSTEX                  Save C[X] in ST, put ST in C[X]
689 F31D1 160          D0=D0+ =oINHS
690 F31D4 14E          C=DAT0 B               Read two nibbles of handshake
691 F31D7 180          D0=D0- =oINHS
692 F31DA 0B           CSTEX                  Put back into ST, restore C[X]
693 F31DC 01           RTN                    Return, carry clear
```

```
                        STITLE Check and set up mailbox
                    ****************************************************************
                    ****************************************************************
                    **
                    ** Name:        CHKSET - Check if this mailbox has been reset
                    ** Name:        CHKST+ - Set up this mailbox after reset
                    **
                    ** Category:    LOCAL
                    **
                    ** Purpose:
                    **      Check if this mailbox has been reset...if so, set up
                    **      device ID and accessory ID
                    **
                    ** Entry:
                    **      DO @ mailbox
                    **
                    ** Exit:
                    **      DO pointing to mailbox
                    **      Carry clear:
                    **        All OK (If mailbox had been reset, it has been set up)
                    **      Carry set:
                    **        Error...P, C[0] are error code
                    **
                    ** Calls:       PUTC,PUTE
                    **
                    ** Uses.......
                    **  Exclusive: A[W],C[W],P
                    **  Inclusive: A[W],C[W],P
                    **
                    ** Stk lvls:   1 (PUTC)(PUTE)
                    **
                    ** Detail:
                    **      Check if RESET bit is set...if not, return, carry clear
                    **      Set IDY timeout = 50 mS
                    **      Set Accessory ID = (mSETAI)
                    **      Set Device ID = (vDEVID)&Cr&Lf
                    **
                    ** History:
                    **
                    **    Date      Programmer              Modification
                    **  --------    ----------    ---------------------------------
                    **  06/03/83        NZ        Added setting IDY timeout to 50ms
                    **  03/16/83        NZ        Added clear of NRD if reset
                    **  02/22/83        NZ        Wrote routine and documentation
                    **
                    ****************************************************************
                    ****************************************************************
F31DE 160   =CHKSET DO=DO+ =oOUTHS
F31E1 1564          C=DATO S                Read into C[S]
F31E5 180           DO=DO- =oOUTHS
F31E8 A46           C=C+C  S                Check if reset
F31EB 500           RTNNC                   If no carry, has NOT been reset
                    *
                    * Need to set device and accessory ID here
                    *
```

```
    749 F31EE AF2          C=0    W
    750 F31F1 15C8         DAT0=C 9             Clear NRD, etc
    751            *
    752 F31F5 20   =CHKST+ P=     0
    753 F31F7 3300         LC(4)  (=mSETIT)+50  Set IDY timeout to 50 msecs
              00
    754 F31FD 8E00         GOSUBL =PUTC
              00
    755 F3203 400          RTNC
    756 F3206 3500         LC(6)  =mSETA1       Set accessory ID length
              0000
    757 F320E 7960         GOSUB  Pute
    758 F3212 400          RTNC
    759 F3215 3500         LC(6)  =mSETAI
              0000
    760 F321D 7A50         GOSUB  Pute          Set accessory ID value
    761 F3221 400          RTNC
    762 F3224 3500         LC(6)  =mSETD1       Set device ID length
              0000
    763 F322C 7B40         GOSUB  Pute
    764 F3230 400          RTNC
    765            *+
    766            *       LCASC  (=vDEVID)+#000A0D*#100000000 xxxx<Cr><Lf>
    767 F3233 3D           NIBHEX 3D
    768 F3235 0000         CON(8) =vDEVID       Value of device ID
              0000
    769 F323D D0A0         NIBHEX D0A000
              00
    770            *+
    771 F3243 AFA          A=C    W             Save in A[W]
    772 F3246 B44  CHKSE1  A=A+1  S             Increment the pointer value
    773 F3249 3500         LC(6)  =mSETDI       Set device ID
              0000
    774 F3251 816          CSRC
    775 F3254 816          CSRC
    776 F3257 AE6          C=A    B             Copy next byte to C[B]
    777 F325A 812          CSLC
    778 F325D AC6          C=A    S             Copy count to C[S]
    779 F3260 812          CSLC                 Now message is set up
    780 F3263 7410         GOSUB  Pute          Send the message
    781 F3267 400          RTNC
    782 F326A 2E           P=     14            Don't alter A[S]
    783 F326C B94          ASR    WP            Get next character
    784 F326F B94          ASR    WP
    785 F3272 20           P=     0
    786 F3274 96C          ?A#0   B             Done yet?
    787 F3277 FC           GOYES  CHKSE1        No...continue
    788 F3279 01           RTN                  Yes...done
    789            *-
    790            *-
    791 F327B 8C00 =Pute   GOLONG =PUTE
              00
    792            *-
    793            *-
    794 F3281 1F00 =D1=DSP D1=(5) =IS-DSP
```

```
          000
795 F3288 01          RTN
796           *-
797           *-
798 F328A 1F00 =D1=DST D1=(5) =DSPSET
          000
799 F3291 01          RTN
800           *-
801           *-
802 F3293 1F00 =D1=DSX D1=(5) =DSPCHX
          000
803 F329A 01          RTN
```

```
                        STITLE Utility routines
                *****************************************************************
                *****************************************************************
                **
                ** Name:        SAVEST - Save status bits in STSAVE
                ** Name:        RESTST - Restore status bits from STSAVE
                **
                ** Category:    SAVUTL
                **
                ** Purpose:
                **      Save or restore status bits in =STSAVE RAM
                **
                ** Entry:
                **      Nothing
                **
                ** Exit:
                **      Status bits saved in/restored from =STSAVE
                **
                ** Calls:       None
                **
                ** Uses.......
                **  Inclusive: STSAVE[2:0]/ST[11:0]
                **
                ** Stk lvls:    1 (internal push)
                **
                ** NOTE: Does not alter carry
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **   --------   ----------    --------------------------------
                **   12/21/82       NZ        Updated documentation
                **
                *****************************************************************
                *****************************************************************
F329C 06        =SAVEST RSTK=C                  Save C[A] on stack
F329E 136               CDOEX                   Save DO in C[A]
F32A1 1B00              DO=(5) =STSAVE
      000
F32A8 0B                CSTEX
F32AA 15C2              DAT0=C 3                Write out the status bits
F32AE 0B        xxxxST  CSTEX
F32B0 134               DO=C                    Restore DO
F32B3 07                C=RSTK                  Restore C[A]
F32B5 01                RTN
                *-
                *-
F32B7 06        =RESTST RSTK=C                  Save C[A] on stack
F32B9 136               CDOEX                   Save DO in C[A]
F32BC 1B00              DO=(5) =STSAVE
      000
F32C3 0B                CSTEX
F32C5 15E2              C=DAT0 3                Read back the status bits
F32C9 64EF              GOTO   xxxxST           Exit (Common code)
                *****************************************************************
```

```
857             ****************************************************************
858             **
859             ** Name:        SAVEDO - Save DO in STMTDO
860             ** Name:        RESTDO - Restore DO from STMTD1
861             ** Name:        SWAPDO - Exchange DO with STMTDO
862             ** Name:        SAVED1 - Save D1 in STMTD1
863             ** Name:        RESTD1 - Restore D1 from STMTD1
864             ** Name:        SAVE1A - Save A[W] in STMTR0
865             ** Name:        REST1A - Restore A[W] from STMTR0
866             ** Name:        SAVE2C - Save C[W] in STMTR1
867             ** Name:        REST2C - Restore C[W] from STMTR1
868             **
869             ** Category:    SAVUTL
870             **
871             ** Purpose:
872             **      Save or restore the value in mainframe STMTxx RAM:
873             **        these go away between statement executions
874             **
875             ** Entry:
876             **      None
877             **
878             ** Exit:
879             **      RESTXX: Restores the register indicated by XX
880             **      SAVEXX: Saves the register indicated by XX
881             **
882             ** Calls:       None
883             **
884             ** Uses.......
885             **  Inclusive: The designated RAM for SAVE, register for REST
886             **
887             ** Stk lvls:    SAVExx: 1
888             ** Stk lvls:    SWAPDO: 2
889             **
890             ** NOTE: Does not alter carry
891             **
892             ** History:
893             **
894             **    Date      Programmer              Modification
895             **  --------    ----------    ---------------------------------
896             **  12/21/82      NZ          Updated documentation
897             **
898             ****************************************************************
899             ****************************************************************
900 F32CD 06    =SAVEDO RSTK=C                Save C[A] on RSTK
901 F32CF 136           CDOEX
902 F32D2 1B00          DO=(5) =STMTDO
        000
903 F32D9 144           DATO=C A
904 F32DC 136   SAVE0r  CDOEX
905 F32DF 07            C=RSTK                Restore C[A] from RSTK
906 F32E1 01            RTN
907             *-
908             *-
909 F32E3 06    =SAVED1 RSTK=C                Save C[A] on RSTK
910 F32E5 137           CD1EX
```

```
911 F32E8 1F00          D1=(5) =STMTD1
          000
912 F32EF 145           DAT1=C A
913 F32F2 137  SAVE1r   CD1EX
914 F32F5 07            C=RSTK              Restore C[A] from RSTK
915 F32F7 01            RTN
916             *-
917             *-
918 F32F9 06   =RESTDO  RSTK=C              Save C[A] on RSTK
919 F32FB 136           CDOEX
920 F32FE 1B00          DO=(5) =STMTDO
          000
921 F3305 146           C=DATO A
922 F3308 63DF          GOTO   SAVEOr
923             *-
924             *-
925 F330C 06   =RESTD1  RSTK=C              Save C[A] on RSTK
926 F330E 137           CD1EX
927 F3311 1F00          D1=(5) =STMTD1
          000
928 F3318 147           C=DAT1 A
929 F331B 66DF          GOTO   SAVE1r
930             *-
931             *-
932 F331F 06   =SWAPDO  RSTK=C              Save C[A] on RSTK
933 F3321 136           CDOEX
934 F3324 06            RSTK=C              Save old DO on RSTK
935 F3326 1B00          DO=(5) =STMTDO      This alters C[A]
          000
936 F332D 146           C=DATO A            Get RAM DO value
937 F3330 136           CDOEX               RAM DO value in DO
938 F3333 07            C=RSTK              Old DO value in C[A] now
939 F3335 136           CDOEX
940 F3338 06            RSTK=C              Now push new DO value
941 F333A 136           CDOEX
942 F333D 1B00          DO=(5) =STMTDO      Get address again
          000
943 F3344 144           DATO=C A            Write out old DO value
944 F3347 07            C=RSTK              Get new DO value from RSTK
945 F3349 629F          GOTO   SAVEOr
946             *-
947             *-
948 F334D 06   =SAVE1A  RSTK=C              Save C[A] on RSTK
949 F334F 136           CDOEX
950 F3352 1B00          DO=(5) =STMTRO
          000
951 F3359 1507          DATO=A W
952 F335D 6E7F          GOTO   SAVEOr
953             *-
954             *-
955 F3361 136  =SAVE2C  CDOEX
956 F3364 06            RSTK=C              Save DO on RSTK
957 F3366 136           CDOEX
958 F3369 1B00          DO=(5) =STMTR1
          000
```

```
  959 F3370 1547          DATO=C W
  960 F3374 136  SAVEOx   CDOEX
  961 F3377 07            C=RSTK                  Restore DO from RSTK
  962 F3379 136           CDOEX
  963 F337C 01            RTN
  964             *-
  965             *-
  966 F337E 06    =REST1A RSTK=C                  Save C[A] on RSTK
  967 F3380 136           CDOEX
  968 F3383 1B00          DO=(5) =STMTRO
          000
  969 F338A 1527          A=DATO W
  970 F338E 6D4F          GOTO   SAVEOr
  971             *-
  972             *-
  973 F3392 136   =REST2C CDOEX                   Get DO into C[A] (Don't care if
  974             *                               C[A] is lost - will be replaced)
  975 F3395 06            RSTK=C                  Save DO on RSTK
  976 F3397 1B00          DO=(5) =STMTR1
          000
  977 F339E 1567          C=DATO W
  978 F33A2 61DF          GOTO   SAVEOx
  979             ********************************************************************
  980             ********************************************************************
  981             **
  982             ** Name:        TSAVDO - Save DO in FUNCDO
  983             ** Name:        TRESDO - Restore DO from FUNCD1
  984             ** Name:        TSWADO - Exchange DO with FUNCDO
  985             ** Name:        TSAVD1 - Save D1 in FUNCD1
  986             ** Name:        TRESD1 - Restore D1 from FUNCD1
  987             ** Name:        TSAV1A - Save A[W] in FUNCRO
  988             ** Name:        TRES1A - Restore A[W] from FUNCRO
  989             ** Name:        TSAV2C - Save C[W] in FUNCR1
  990             ** Name:        TRES2C - Restore C[W] from FUNCR1
  991             **
  992             ** Category:    SAVUTL
  993             **
  994             ** Purpose:
  995             **      Save or restore the value in mainframe FUNCxx RAM:
  996             **        these go away during function executions
  997             **
  998             ** Entry:
  999             **      None
 1000             **
 1001             ** Exit:
 1002             **      TRESxx: Restores the register indicated by xx
 1003             **      TSAVxx: Saves the register indicated by xx
 1004             **
 1005             ** Calls:       None
 1006             **
 1007             ** Uses.......
 1008             **  Inclusive: The designated RAM for TSAV, register for TRES
 1009             **
 1010             ** Stk lvls:   TSAVxx: 1
 1011             ** Stk lvls:   TSWAD1: 2
```

```
1012                    **
1013                    ** NOTE: Does not alter carry
1014                    **
1015                    ** History:
1016                    **
1017                    **    Date     Programmer          Modification
1018                    **  --------   ----------   --------------------------------
1019                    **  12/21/82      NZ        Updated documentation
1020                    **
1021                    ********************************************************************
1022                    ********************************************************************
1023 F33A6 06          =TSAVD0 RSTK=C              Save C[A] on RSTK
1024 F33A8 136                 CD0EX
1025 F33AB 1B00                D0=(5) =FUNCD0
           000
1026 F33B2 144                 DAT0=C A
1027 F33B5 136         TSAV0r  CD0EX
1028 F33B8 07                  C=RSTK               Restore C[A] from RSTK
1029 F33BA 01                  RTN
1030                    *-
1031                    *-
1032 F33BC 06          =TSAVD1 RSTK=C              Save C[A] on RSTK
1033 F33BE 137                 CD1EX
1034 F33C1 1F00                D1=(5) =FUNCD1
           000
1035 F33C8 145                 DAT1=C A
1036 F33CB 137         TSAV1r  CD1EX
1037 F33CE 07                  C=RSTK               Restore C[A] from RSTK
1038 F33D0 01                  RTN
1039                    *-
1040                    *-
1041 F33D2 06          =TRESD0 RSTK=C              Save C[A] on RSTK
1042 F33D4 136                 CD0EX
1043 F33D7 1B00                D0=(5) =FUNCD0
           000
1044 F33DE 146                 C=DAT0 A
1045 F33E1 63DF                GOTO   TSAV0r
1046                    *-
1047                    *-
1048 F33E5 06          =TRESD1 RSTK=C              Save C[A] on RSTK
1049 F33E7 137                 CD1EX
1050 F33EA 1F00                D1=(5) =FUNCD1
           000
1051 F33F1 147                 C=DAT1 A
1052 F33F4 66DF                GOTO   TSAV1r
1053                    *-
1054                    *-
1055 F33F8 06          =TSWAD1 RSTK=C              Save C[A] on RSTK
1056 F33FA 137                 CD1EX
1057 F33FD 06                  RSTK=C               Save old D1 on RSTK
1058 F33FF 1F00                D1=(5) =FUNCD1       This alters C[A]
           000
1059 F3406 147                 C=DAT1 A             Get RAM D1 value
1060 F3409 137                 CD1EX                RAM D1 value in D1
1061 F340C 07                  C=RSTK               Old D1 value in C[A] now
```

```
1062 F340E 137          CD1EX
1063 F3411 06           RSTK=C              Now push new D1 value
1064 F3413 137          CD1EX
1065 F3416 1F00         D1=(5) =FUNCD1      Get address again
           000
1066 F341D 145          DAT1=C A            Write out old D1 value
1067 F3420 07           C=RSTK              Get new D1 value from RSTK
1068 F3422 137          CD1EX
1069 F3425 07           C=RSTK              Recall old C[A]
1070 F3427 01           RTN
1071             *-
1072             *-
1073 F3429 136  =TSAV2C CDOEX
1074 F342C 06           RSTK=C              Save DO on RSTK
1075 F342E 136          CDOEX
1076 F3431 1B00         DO=(5) =FUNCR1
           000
1077 F3438 1547         DATO=C W
1078 F343C 136  TSAVOx  CDOEX
1079 F343F 07           C=RSTK              Restore DO from RSTK
1080 F3441 136          CDOEX
1081 F3444 01           RTN
1082             *-
1083             *-
1084 F3446 136  =TRES2C CDOEX               Get DO into C[A] (Don't care if
1085             *                          C[A] is lost - will be replaced)
1086 F3449 06           RSTK=C              Save DO on RSTK
1087 F344B 1B00         DO=(5) =FUNCR1
           000
1088 F3452 1567         C=DATO W
1089 F3456 65EF         GOTO   TSAVOx
1090             ************************************************************
1091             ************************************************************
1092             **
1093             ** Name:       SAVSTS - Save RSTK levels, Status bits, D[A]
1094             **
1095             ** Category:   SAVUTL
1096             **
1097             ** Purpose:
1098             **      Save 6 stack levels and status bits AND D[A] in SNAPBF
1099             **
1100             ** Entry:
1101             **      C[A] is first stack level
1102             **
1103             ** Exit:
1104             **      P=0, stack levels saved in =SNAPBF
1105             **      Carry clear
1106             **
1107             ** Calls:      None
1108             **
1109             ** Uses.......
1110             **  Inclusive: B[A],C[A],DO,P,SNAPBF[37:0]
1111             **
1112             ** Stk lvls:   (-6) (Saved in SNAPBF)
1113             **
```

```
1114                ** History:
1115                **
1116                **    Date     Programmer             Modification
1117                **  --------   ----------   --------------------------------
1118                **  12/21/82      NZ         Updated documentation
1119                **
1120                ****************************************************************
1121                ****************************************************************
1122 F345A 2B      =SAVSTS P=     16-5           Save 5 more levels
1123 F345C 1B00            DO=(5) =SNAPBF        Snap buffer
           000
1124 F3463 144     =SAVST+ DAT0=C A              Write out first address
1125 F3466 164             DO=DO+ 5
1126 F3469 09              C=ST
1127 F346B 15C2            DAT0=C 3              Save status bits
1128 F346F 162             DO=DO+ 3
1129 F3472 07              C=RSTK                Pop calling address
1130 F3474 D5              B=C    A              Save calling address in B[A]
1131 F3476 07      SAVSTs  C=RSTK                Pop a level
1132 F3478 144             DAT0=C A              Save it in SNAPBF
1133 F347B 164             DO=DO+ 5
1134 F347E 0C              P=P+1
1135 F3480 55F             GONC   SAVSTs         If no carry, not done yet
1136 F3483 D9              C=B    A              Recall calling address...
1137 F3485 06              RSTK=C                ...push back on stack...
1138 F3487 DB              C=D    A              ...SAVE D[A]...
1139 F3489 144             DAT0=C A
1140 F348C 03              RTNCC                 ...and return, carry clear
1141                ****************************************************************
1142                ****************************************************************
1143                **
1144                ** Name:       RESSTS - Restore RSTK lvls, D[A], and statuses
1145                **
1146                ** Category:   SAVUTL
1147                **
1148                ** Purpose:
1149                **      Restore status, 6 stack levels, and D[A] from =SNAPBF
1150                **
1151                ** Entry:
1152                **      Nothing
1153                **
1154                ** Exit:
1155                **      P=0, last stack level in C[A]
1156                **      Carry clear
1157                **
1158                ** Calls:      None
1159                **
1160                ** Uses.......
1161                **  Inclusive: B[A],C[A],DO,P
1162                **
1163                ** Stk lvls:   (+6) (Restores RSTK levels from SNAPBF)
1164                **
1165                ** History:
1166                **
1167                **    Date     Programmer             Modification
```

```
1168                ** --------  ----------  --------------------------------
1169                ** 12/21/82      NZ          Updated documentation
1170                **
1171                ****************************************************************
1172                ****************************************************************
1173 F348E 2B      =RESSTS P=      16-5           # of levels to restore -1
1174 F3490 1B00            DO=(5) (=SNAPBF)+(6*5)+3 6 pointers @ 5 nibs+ 3 status
          000
1175 F3497 146             C=DATO A
1176 F349A D7              D=C    A               Restore D[A]
1177 F349C 07      =RESST+ C=RSTK                 Pop calling address
1178 F349E D5              B=C    A               Save in B[A]
1179 F34A0 184     RESSTs  DO=DO- 5               Predecrement the data pointer
1180 F34A3 146             C=DATO A               Read the pointer
1181 F34A6 06              RSTK=C                 Push address onto stack
1182 F34A8 0C              P=P+1
1183 F34AA 55F             GONC   RESSTs          Loop back for next pointer
1184                *
1185                * Now fetch status bits and last stack level
1186                *
1187 F34AD 182             DO=DO- 3
1188 F34B0 146             C=DATO A               Read status bits
1189 F34B3 0A              ST=C                   Push into status bits
1190 F34B5 D9              C=B    A
1191 F34B7 06              RSTK=C                 Push calling address onto stack
1192 F34B9 184             DO=DO- 5
1193 F34BC 146             C=DATO A               Read last level
1194 F34BF 03              RTNCC
```

```
1195                    STITLE HPIL error message driver
1196            ****************************************************************
1197            ****************************************************************
1198            **
1199            ** Name:        ERROR - Error driver routine
1200            ** Name:        ERRORX - Error driver for execution errors
1201            ** Name:        ERRORP - Error driver for parse errors
1202            ** Name:        ERRORR - Error driver for parse (no RESPTR)
1203            **
1204            ** Category:    PILUTL
1205            **
1206            ** Purpose:
1207            **      ERRORX is execute error - jumps to mferr
1208            **      ERRORP is parse error - jumps to PARERR
1209            **      ERRORR is parse error - jumps to PARERR, no RESPTR
1210            **
1211            ** Entry:
1212            **      P contains the error type:
1213            **              0: Parse error (Type in C[0])
1214            **              1: Tape error (Type in C[0])
1215            **              2: HPIL error (Type in C[0])
1216            **              3: <undefined>
1217            **              4: Aborted
1218            **              5: Invalid Device Spec
1219            **              6: Non-numeric data
1220            **              7: <undefined>
1221            **              8: Out of range value
1222            **              9: No Mailbox
1223            **             10: <undefined>
1224            **             11: Insufficient Memory
1225            **             12: RESTORE IO needed
1226            **             13: <undefined>
1227            **             14: <undefined>
1228            **             15: <undefined>
1229            **
1230            ** Exit:
1231            **      ERRORX, ERRORP, and ERRORR return to the mainframe
1232            **      The error # is in C[B], P=0, C[3:2] is HP-IL LEX id
1233            **      Carry set
1234            **
1235            ** Calls:       GETMBX,ATNCHK,GETERR
1236            **
1237            ** Uses.......
1238            **  Inclusive: C[W],D0,P
1239            **
1240            ** Stk lvls:    2 (GETERR) {ERRORX, ERRORP, ERRORR use 3}
1241            **
1242            ** History:
1243            **
1244            **    Date      Programmer              Modification
1245            **  --------    ----------    ------------------------------
1246            **  01/24/84       NZ         Check P=  =eABORT after call to
1247            **                            GETERR (if so, need to jump to a
1248            **                            different place)
1249            **  12/21/82       NZ         Updated documentation
```

```
1250                **
1251                ************************************************************
1252                ************************************************************
1253 F34C1 7020 =ERRORX GOSUB  ERROR          Set up the error message
1254 F34C5 8C00         GOLONG =bSERR         (Jump to BSERR in mainframe)
           00
1255                *-
1256                *-
1257 F34CB 854  =ERRORR ST=1   4              Don't restore ntoken
1258 F34CE 80F0 =ERRORP CPEX   0              Put error # in C[0]
1259 F34D2 20           P=     =ePARSE        Parse error
1260 F34D4 7D00         GOSUB  ERROR          Set up the error message
1261 F34D8 84A  =ERROR! ST=0   10             Clear implied LET flag...
1262 F34DB 136          CDOEX                 Error # in DO[3:0]
1263 F34DE 8D00         GOVLNG =PARERR        ...and jump to error routine
           000
1264                *-
1265                *-
1266 F34E5 890  =ERROR  ?P=    =ePARSE        Is this a parse error?
1267 F34E8 23           GOYES  ERROR1         Yes...error subclass
1268 F34EA 890          ?P=    =eTAPE         Tape error?
1269 F34ED D2           GOYES  ERROR1         Yes...error subclass
1270 F34EF 890          ?P=    =ePIL          HPIL mailbox error?
1271 F34F2 82           GOYES  ERROR1         Yes...error subclass
1272 F34F4 880          ?P#    =eABORT        "Aborted"?
1273 F34F7 D1           GOYES  ERRORO         No...set up the message
1274                *
1275                * Aborted out...try to check status
1276                *
1277 F34F9 7000         GOSUB  =GETMBX        Get the last mailbox used
1278 F34FD 8E00         GOSUBL =ATNCHK        Check if ATTN key hit twice
           00
1279 F3503 401          GOC    ERRORO         Yes...abort out
1280 F3506 7E11         GOSUB  Geterr         Get the error message
1281 F350A 570          GONC   ERROR-         No error...say "Aborted"
1282 F350D 880          ?P#    =eABORT        Error...is it "Aborted"?
1283 F3510 A0           GOYES  ERROR1         No...set up the message
1284                *
1285 F3512 20   ERROR-  P=     =eABORT        "Aborted"
1286                *
1287                * P>ePIL...set C[0]=P, C[1]=ePIL+1
1288                *
1289 F3514 80C0 ERRORO  C=P    0              Put error # in C[0]
1290 F3518 20           P=     (=ePIL)+1
1291 F351A 80C1 ERROR1  C=P    1              Error class --> C[1]
1292 F351E 22           P=     2
1293 F3520 3100         LC(2)  =LEXPIL
1294 F3524 20           P=     0
1295 F3526 02           RTNSC
```

```
1296                    STITLE File spec execute handler
1297            ****************************************************************
1298            ****************************************************************
1299            **
1300            ** Name:        FILSPx - File spec execution routine
1301            **
1302            ** Category:    POLL
1303            **
1304            ** Purpose:
1305            **      File spec execution poll handler
1306            **
1307            ** Entry:
1308            **      ST(=sSTK) indicates whether this is literal/string
1309            **      P=0
1310            **      If literal:
1311            **        STMTDO points to start of file spec
1312            **      If string:
1313            **        TASTK (=AVMEME) points to the string header in RAM
1314            **
1315            ** Exit:
1316            **      Carry XM
1317            **      ----- --
1318            **        0    0: Handled:A=first 8,R0=last 2 chars of name;
1319            **                        D[S]=8; D[X]=loop address; ST8=1
1320            **                        D[3]:bit 3 is don't fill in name,
1321            **                             bit 2 is Acc ID=16 device
1322            **                        R3=Device ID/Volume lbl; R2=output
1323            **                        from SETUP (R2[14]=8!)
1324            **                        ST[8]=1 (not simple filename)
1325            **        0    1: Not handled: Nothing (DO restored by POLL)
1326            **        1    X: Error: C[3:0] is error code for mferr*
1327            **
1328            ** Calls:       SAVEST,D1@AVE,POP1S,D1=SDO,GETPI+,CHKMAS,ASLC4,
1329            **              RESTST,TRESDO
1330            **
1331            ** Uses.......
1332            **  Exclusive: A,  C,D,R0,   R2,R3,   D0,D1,P
1333            **  Inclusive: A,B,C,D,R0,R1,R2,R3,R4,D0,D1,P,FUNCxx,STMTR1,
1334            **              STMTD1[3:0],ST[sDevOK]
1335            **  SETS ST(8) if handled
1336            **
1337            ** Stk lvls:    6 (GETPI+)
1338            **
1339            ** History:
1340            **
1341            **    Date      Programmer              Modification
1342            **  --------    ----------    ---------------------------------
1343            **  05/31/83       NZ         Reworked acc ID check to take
1344            **                            less code by removing check for
1345            **                            mass storage, NOT Acc ID=16
1346            **  05/11/83       NZ         Added check of accessory ID to
1347            **                            return with a bit indicating mass
1348            **                            storage - Acc ID=16, also able to
1349            **                            properly indicate "FILL" bit now
1350            **  03/17/83       NZ         Modified code around GETPI+ to
```

```
1351                **                              match new entry/exit conditions
1352                **  02/11/83      NZ            Added LOOP check for device type
1353                **  12/21/82      NZ            Updated documentation
1354                **
1355                ****************************************************************
1356                ****************************************************************
1357                *
1358                * Necessary to save status...GETPI+ saves them only if calls
1359                * to EXPEXC are needed for an expression
1360                *
1361 F3528 707D =FILSPx GOSUB  SAVEST         Save status bits in =STSAVE
1362 F352C 860          ?ST=0  =sSTK          Is this a literal in memory?
1363 F352F E1           GOYES  FILSx1         Yes...recall start
1364                *
1365                * This is a string expression (already on the stack)
1366                *
1367 F3531 8E00          GOSUBL =D1@AVE        (TASTK=AVMEME=MTHSTK)
           00
1368 F3537 8F00          GOSBVL =POP1S         Pop the string
           000
1369                *
1370                * Now D1 @ start of string, A[A] is length
1371                *
1372 F353E 137           CD1EX
1373 F3541 D7            D=C    A              Temp save start in D[A]
1374 F3543 C2            C=C+A  A
1375 F3545 DF            CDEX   A              Now end in D[A], start in C[A]
1376 F3547 137           CD1EX                 D1 points to start of string
1377 F354A 5D1           GONC   FILSx2         Go always
1378                *-
1379                *-
1380 F354D 8E00 FILSx1   GOSUBL =D1=SDO        Set D1 @ STMTDO
           00
1381 F3553 143           A=DAT1 A
1382 F3556 130           D0=A                  Point D0 to the start of spec
1383 F3559 14A           A=DAT0 B              If first character is tLITRL,
1384 F355C 3100          LC(2)  =tLITRL          skip it
1385 F3560 966           ?A#C   B
1386 F3563 50            GOYES  FILSx2         Not tLITRL...go on
1387 F3565 161           D0=D0+ 2              tLITRL...skip over it
1388                *
1389                * Now D0 @ start of literal/D1 at start of string
1390                *
1391 F3568 8E00 FILSx2   GOSUBL =GETPI+        Get the file name and device spec
           00
1392 F356E 427           GOC    FILSPs         Not mine...don't handle it
1393                *
1394                * Now B,D have everything needed to find the device again
1395                *
1396                * Clear unused bits in D[M]
1397                *
1398 F3571 AD3           D=0    M              Clear D[4:3] without changing D[S]
1399                *
1400                * Check if file spec was "" or "*" (if so, don't handle it)
1401                *
```

```
1402 F3574 96F          ?D#0   B             Not LOOP or NULL or "" or *
1403 F3577 C0           GOYES  FILSx.
1404              *
1405              * Check that this is NOT "LOOP"
1406              *
1407 F3579 2F           P=     15
1408 F357B 300          LC(1)  =DsLoop       Check if LOOP
1409 F357E 947          ?D#C   S             LOOP?
1410 F3581 A6           GOYES  FILSPm        No...don't handle it
1411              *
1412              * This is "LOOP"...not Acc ID=16 or mass storage, don't fill
1413              * name (Carry is CLEAR for LOOP)
1414              *
1415              * Set up for the mainframe to be able to save the device info
1416              *
1417 F3583 2E     FILSx.  P=     14
1418 F3585 308          LC(1)  8             Set device code=8 (HPIL)
1419 F3588 10A          R2=C                 Save output from SETUP in R2
1420 F358B AF9          C=B    W
1421 F358E 10B          R3=C                 Save device ID/volume label in R3
1422 F3591 512          GONC   FILSx1        Go if "LOOP" was specified
1423              *
1424              * First check what the accessory ID is...
1425              *
1426 F3594 8E00         GOSUBL =CHKMAS       Check if mass storage
          00
1427 F359A 4A0          GOC    FILSx?        Either error or not Acc ID=16
1428 F359D 23           P=     3
1429 F359F 304          LC(1)  4             This is Acc ID=16, fill in name
1430 F35A2 551          GONC   FILSx#        Go always
1431              *-
1432              *-
1433              *
1434              * Check if the accessory ID is "MASS STORAGE"
1435              *
1436 F35A5 880  FILSx?  ?P#    =ePIL
1437 F35A8 15           GOYES  FILSPe        Error...not HPIL error
1438 F35AA 80F0         CPEX   0             First check if Device Type error
1439 F35AE 880          ?P#    =eDTYPE
1440 F35B1 44           GOYES  FILSPE        Error
1441              *
1442              * This IS a device type error...
1443              *
1444 F35B3 23   FILSx1  P=     3
1445 F35B5 308          LC(1)  8
1446 F35B8 A87  FILSx#  D=C    P             Set the "Don't fill filename" bit
1447 F35BB 2F           P=     15
1448 F35BD 308          LC(1)  8
1449 F35C0 20           P=     0
1450              *
1451              * Device 8 is HP-IL
1452              *
1453 F35C2 AC7          D=C    S             First 8 chars in A[W]
1454 F35C5 114          A=R4
1455 F35C8 8E00         GOSUBL =ASLC4        Last 2 chars in A[3:0]
```

```
            00
  1456 F35CE 120          AROEX               First 8 chars in A, last 2 in R0
  1457              *
  1458              * Restore the caller's status first
  1459              *
  1460 F35D1 72EC         GOSUB  RESTST
  1461              *
  1462              * Now restore D0 (PC) following the device spec
  1463              *
  1464 F35D5 79FD         GOSUB  TRESD0       (Saved by GETPI+)
  1465              *
  1466              * ST[8] means this is not a simple filename...
  1467              *
  1468 F35D9 858          ST=1   8
  1469 F35DC 821          XM=0                Be sure XM is zero - handled
  1470 F35DF 03           RTNCC               Return (Handled, OK)
  1471              *-
  1472              *-
  1473 F35E1 890  FILSPs  ?P=    =eNORAM      Did I run out of memory?
  1474 F35E4 51           GOYES  FILSPe       Yes...error
  1475 F35E6 870          ?ST=1  =sDevOK      Was the device spec OK?
  1476 F35E9 01           GOYES  FILSPe       Yes...loop error
  1477 F35EB 78CC FILSPm  GOSUB  RESTST       Restore status bits from =STSAVE
  1478 F35EF 21   DIDST1  P=     1
  1479 F35F1 0D           P=P-1               Clear carry, P=0
  1480 F35F3 00           RTNSXM              Return carry clear, XM set
  1481              *-
  1482              *-
  1483 F35F5 80F0 FILSPE  CPEX   0
  1484 F35F9 6BEE FILSPe  GOTO   ERROR        Return with C[3:0]->error #,RTNSC
```

```
1485                    STITLE Store device ID handler
1486                ******************************************************************
1487                ******************************************************************
1488                **
1489                ** Name:       hDIDST - Store device ID info (from R2,R3)
1490                **
1491                ** Category:   POLL
1492                **
1493                ** Purpose:
1494                **      Handler for device ID storage (D1 @ destination point)
1495                **
1496                ** Entry:
1497                **      R2 contains C[W] from SETUP
1498                **      (R2[14] is the device code from FILSPx)
1499                **      R3 contains the device ID/volume label
1500                **
1501                ** Exit:
1502                **      P=0
1503                **      Carry clear:
1504                **        XM=0: Device ID saved @ D1
1505                **        XM=1: Not HPIL (No response)
1506                **      (If error, takes direct error jump to ERRORX)
1507                **
1508                ** Calls:      SNAPRS,SAVEIT
1509                **
1510                ** Uses.......
1511                **  Exclusive:   B,C,              P
1512                **  Inclusive: A,B,C,D,R2,R3,D0,D1,P (If not handled, only C,P)
1513                **
1514                ** Stk lvls:   4 (SAVEIT)
1515                **
1516                ** History:
1517                **
1518                **    Date      Programmer             Modification
1519                **  --------    ----------    ---------------------------------
1520                **  01/24/84       NZ         Moved DIDST1 into FILSPx to make
1521                **                            room for a GOLONG (needed 2 nibs)
1522                **  04/15/83       NZ         Moved first SNAPRS call to save D
1523                **                            in case not handled (FPOLL needs
1524                **                            D[A] to be around if not handled)
1525                **  04/01/83       SC         Changed to FPOLL, added SNAPRS
1526                **                            calls to set up pointers
1527                **  12/21/82       NZ         Updated documentation
1528                **
1529                ******************************************************************
1530                ******************************************************************
1531 F35FD          =hDIDST
1532 F35FD 11A              C=R2
1533 F3600 80DE             P=C     14
1534 F3604 888              ?P#     8              Is this an HPIL assignment?
1535 F3607 8E               GOYES   DIDST1         No...leave it alone
1536 F3609 8E00             GOSUBL  =sNAPRS        Restore D1 from save area
          00
1537 F360F 11B              C=R3
1538 F3612 AF5              B=C     W
```

```
1539 F3615 11A          C=R2
1540 F3618 8E00         GOSUBL =SAVEIT      Save the information @ (D1)
           00
1541 F361E 821          XM=0                Make sure XM=0
1542 F3621 500          RTNNC               If no carry, all OK...done
1543 F3624 6C9E         GOTO   ERRORX       If carry, error exit
1544            *-
1545            *-
1546 F3628 8C00 Geterr  GOLONG =GETERR      Jump to get error message
           00
1547 F362E              END
```

```
 ASLC4   Ext                  -  1455
 ATNCHK  Ext                  -  1278
 ATNFLG  Ext                  -   180
 BDISPJ  Ext                  -   279
 CHKASN  Ext                  -   271
 CHKMAS  Ext                  -  1426
 CHKSE1  Abs  995910 #F3246   -   772   787
=CHKSET  Abs  995806 #F31DE   -   741   602
=CHKST+  Abs  995829 #F31F5   -   752    87
 CHKSTS  Ext                  -   369
 CSLC5   Ext                  -   431
 CSRC5   Ext                  -   433
=D1=DSP  Abs  995969 #F3281   -   794    96   269
=D1=DST  Abs  995978 #F328A   -   798   113   244   509
=D1=DSX  Abs  995987 #F3293   -   802   273   287
 D1=SDO  Ext                  -  1380
 D1@AVE  Ext                  -  1367
 DIDST1  Abs  996847 #F35EF   -  1478  1535
 DSPCHX  Ext                  -   802
 DSPSET  Ext                  -   435   798
 DispOK  Ext                  -   247
 DsLoop  Ext                  -  1408
=ERROR   Abs  996581 #F34E5   -  1266  1253  1260  1484
=ERROR!  Abs  996568 #F34D8   -  1261
 ERROR-  Abs  996626 #F3512   -  1285  1281
 ERROR0  Abs  996628 #F3514   -  1289  1273  1279
 ERROR1  Abs  996634 #F351A   -  1291  1267  1269  1271  1283
=ERRORP  Abs  996558 #F34CE   -  1258
=ERRORR  Abs  996555 #F34CB   -  1257
=ERRORX  Abs  996545 #F34C1   -  1253  1543
 Except  Ext                  -   617
 FIBOFF  Ext                  -   432
 FILSPE  Abs  996853 #F35F5   -  1483  1440
 FILSPe  Abs  996857 #F35F9   -  1484  1437  1474  1476
 FILSPm  Abs  996843 #F35EB   -  1477  1410
 FILSPs  Abs  996833 #F35E1   -  1473  1392
=FILSPx  Abs  996648 #F3528   -  1361
 FILSx#  Abs  996792 #F35B8   -  1446  1430
 FILSx.  Abs  996739 #F3583   -  1417  1403
 FILSx1  Abs  996685 #F354D   -  1380  1363
 FILSx2  Abs  996712 #F3568   -  1391  1377  1386
 FILSx?  Abs  996773 #F35A5   -  1436  1427
 FILSxl  Abs  996787 #F35B3   -  1444  1422
 FNDMBX  Ext                  -    81   367   589
 FUNCD0  Ext                  -  1025  1043
 FUNCD1  Ext                  -  1034  1050  1058  1065
 FUNCR1  Ext                  -  1076  1087
 GETERR  Ext                  -    85  1546
=GETHSS  Abs  995791 #F31CF   -   688   591
 GETMBX  Ext                  -  1277
 GETPI+  Ext                  -  1391
 GETST-  Ext                  -   605
 Geterr  Abs  996904 #F3628   -  1546  1280
 I/OALL  Ext                  -    70
 I/ORES  Ext                  -   302
```

```
 I/ores  Abs  995371 #F302B -   302   232   265   268
 IS-DSP  Ext                -   258   441   794
 KEYPTR  Ext                -   641
 LEXPIL  Ext                -  1293
 LOOPST  Ext                -   111   347   499
 LoopOK  Ext                -   512
 MBOX^   Ext                -   583   649
 Offed   Ext                -   502
 PARERR  Ext                -  1263
 PILCN1  Abs  995233 #F2FA1 -   240   233
=PILCNF  Abs  995217 #F2F91 -   230
 PILCS0  Abs  995031 #F2ED7 -    65
 PILCS3  Abs  995066 #F2EFA -    80    89
 PILCS4  Abs  995101 #F2F1D -    92    82
=PILCST  Abs  995031 #F2ED7 -    59   238
 PILM05  Abs  995577 #F30F9 -   504   503
=PILMLP  Abs  995559 #F30E7 -   499
 PILMRC  Abs  995601 #F3111 -   515   505
 PILP01  Abs  995419 #F305B -   366   388
 PILP02  Abs  995459 #F3083 -   387   370   376
 PILP03  Abs  995465 #F3089 -   392   353   361   368
=PILPOF  Abs  995378 #F3032 -   343
 PILS00  Abs  995627 #F312B -   576   575
 PILS07  Abs  995605 #F3115 -   566   571   577
 PILS20  Abs  995652 #F3144 -   588   596
 PILS23  Abs  995673 #F3159 -   594   618   626   631   637
 PILS50  Abs  995766 #F31B6 -   647   590   608
 PILS9+  Abs  995787 #F31CB -   653   566
=PILSRQ  Abs  995609 #F3119 -   569
=PILWKP  Abs  995353 #F3019 -   290
 PILWKs  Abs  995357 #F301D -   294   259   261
=PILWNK  Abs  995176 #F2F68 -   166
 PILWNx  Abs  995213 #F2F8D -   186   168   179
 PILxxx  Abs  995334 #F3006 -   282   277
 POP1S   Ext                -  1368
 PUTC    Ext                -   754
 PUTC+   Ext                -   382
 PUTCN   Ext                -   604
 PUTE    Ext                -   791
=Pute    Abs  995963 #F327B -   791   757   760   763   780
 REQS10  Abs  995714 #F3182 -   611   606
 REQS30  Abs  995727 #F318F -   621   616
 REQSER  Abs  995680 #F3160 -   602   593
=RESST+  Abs  996508 #F349C -  1177
=RESSTS  Abs  996494 #F348E -  1173   651
 RESSTs  Abs  996512 #F34A0 -  1179  1183
=REST1A  Abs  996222 #F337E -   966
=REST2C  Abs  996242 #F3392 -   973
=RESTD0  Abs  996089 #F32F9 -   918
=RESTD1  Abs  996108 #F330C -   925
 RESTOR  Ext                -   253
=RESTRT  Abs  995469 #F308D -   429   343
 RESTRs  Abs  995526 #F30C6 -   448   442   443   444
=RESTST  Abs  996023 #F32B7 -   850  1460  1477
 RESTs4  Abs  995552 #F30E0 -   461   455   457
```

```
=RTNCCX  Abs  995170 #F2F62 -   129   186   291   392   515   653
 RTNCCx  Abs  995353 #F3019 -   291   272   276
 SAVEOr  Abs  996060 #F32DC -   904   922   945   952   970
 SAVEOx  Abs  996212 #F3374 -   960   978
=SAVE1A  Abs  996173 #F334D -   948
 SAVE1r  Abs  996082 #F32F2 -   913   929
=SAVE2C  Abs  996193 #F3361 -   955
=SAVED0  Abs  996045 #F32CD -   900
=SAVED1  Abs  996067 #F32E3 -   909
 SAVEIT  Ext                -  1540
=SAVEST  Abs  995996 #F329C -   839  1361
=SAVST+  Abs  996451 #F3463 -  1124
=SAVSTS  Abs  996442 #F345A -  1122   582
 SAVSTs  Abs  996470 #F3476 -  1131  1135
 SNAPBF  Ext                -  1123  1174
 STMTD0  Ext                -   902   920   935   942
 STMTD1  Ext                -   911   927
 STMTR0  Ext                -   950   968
 STMTR1  Ext                -   958   976
 STSAVE  Ext                -   841   852
=SWAPD0  Abs  996127 #F331F -   932
 TERCHR  Ext                -   123
=TRES2C  Abs  996422 #F3446 -  1084
=TRESD0  Abs  996306 #F33D2 -  1041  1464
=TRESD1  Abs  996325 #F33E5 -  1048
 TSAVOr  Abs  996277 #F33B5 -  1027  1045
 TSAVOx  Abs  996412 #F343C -  1078  1089
 TSAV1r  Abs  996299 #F33CB -  1036  1052
=TSAV2C  Abs  996393 #F3429 -  1073
=TSAVD0  Abs  996262 #F33A6 -  1023
=TSAVD1  Abs  996284 #F33BC -  1032
=TSWAD1  Abs  996344 #F33F8 -  1055
 WNKOO   Abs  995194 #F2F7A -   178   177
 bPILAI  Ext                -   267
 bPILSV  Ext                -    69   231
 bSERR   Ext                -  1254
 bSTMXQ  Ext                -   264
 eABORT  Ext                -   607  1272  1282  1285
 eDTYPE  Ext                -  1439
 eNORAM  Ext                -  1473
 ePARSE  Ext                -  1259  1266
 ePIL    Ext                -  1270  1290  1436
 eTAPE   Ext                -  1268
 flDORM  Ext                -   635
 flPDWN  Ext                -   359
=hDIDST  Abs  996861 #F35FD -  1531
 hsRQSR  Ext                -   592
 mPDLOP  Ext                -   381
 mSETAI  Ext                -   759
 mSETAl  Ext                -   756
 mSETDI  Ext                -   773
 mSETDl  Ext                -   762
 mSETIT  Ext                -   753
 mSTSTC  Ext                -   603
 oINHS   Ext                -   689   691
```

oOUTHS	Ext			-	741	743
sCONTR	Ext			-	375	
sDATAV	Ext			-	625	
sDIAsr	Ext			-	176	574
sDevOK	Ext			-	1475	
sFLAG?	Ext			-	360	636
sINTR	Ext			-	615	
sNAPRS	Ext			-	1536	
sRMOTE	Ext			-	630	
sSTK	Ext			-	1362	
tLITRL	Ext			-	1384	
vDEVID	Ext			-	768	
xxxxST	Abs	996014	#F32AE	-	844	855

Input Parameters

  Source file name is NZ&BIF::MS

  Listing file name is NZ/BIF:TI:ML

  Object file name is NZ%BIF:TI:MS

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *
 2            *
 3            *       N   N  ZZZZZ   &     III  00000  BBBB
 4            *       N   N      Z  & &     I   0   0  B   B
 5            *       NN  N     Z   & &     I   0   0  B   B
 6            *       N N N    Z     &      I   0   0  BBBB
 7            *       N  NN   Z     & & &   I   0   0  B   B
 8            *       N   N  Z      & &     I   0   0  B   B
 9            *       N   N  ZZZZZ   && &  III  00000  BBBB
10            *
11                    TITLE  I/O Buffer routines <830927.1450>
12 F362E              ABS    #F362E        TI%HP6 address (fixed)
13                   ****************************************************************
14                   ****************************************************************
15                   **
16                   ** Name:       I/OFSC - Find a scratch I/O buffer (#C00->#FFF)
17                   **
18                   ** Category:   BUFUTL
19                   **
20                   ** Purpose:
21                   **      File I/O scratch buffer (Return ID of first unused
22                   **      buffer)
23                   **
24                   ** Entry:
25                   **      Nothing
26                   **
27                   ** Exit:
28                   **      P=0
29                   **      Carry clear: C[X] is buffer ID
30                   **      Carry set: no buffer available (C[X]=0)
31                   **
32                   ** Calls:      I/OFND
33                   **
34                   ** Uses.......
35                   **  Inclusive: A[W],C[X],D1,P
36                   **
37                   ** Stk lvls:   1 (I/OFND)
38                   **
39                   ** History:
40                   **
41                   **    Date     Programmer             Modification
42                   **  --------   ----------    ---------------------------------
43                   **  09/27/83      NZ         Changed documentation to reflect
44                   **                           current routine (IOFSCR)
45                   **  01/04/83      NZ         Updated documentation
46                   **
47                   ****************************************************************
48                   ****************************************************************
49 F362E 20          =I/OFSC P=     0
50 F3630 8D00        000            GOVLNG =IOFSCR
51 F3637              END
```

```
=I/OFSC  Abs  996910 #F362E -    49
 IOFSCR  Ext                -    50
```

Input Parameters

  Source file name is NZ&IOB::MS

  Listing file name is NZ/IOB:TI:ML::-1

  Object file name is NZ%IOB:TI:MS::-1

                                    111111
                          0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
                *
                *       N   N  ZZZZZ   &     DDDD    SSS   PPPP
                *       N   N      Z  & &    D  D  S   S  P   P
                *       NN  N     Z   & &    D  D  S      P   P
                *       N N N    Z     &     D  D   SSS   PPPP
                *       N  NN   Z     & & &   D  D      S  P
                *       N   N  Z      &  &    D  D  S   S  P
                *       N   N  ZZZZZ   && &  DDDD    SSS   P
                *
                *
                        TITLE  Display driver <831108.0941>
F3637           ABS    #F3637        TI%HP6 address (fixed)
                ****************************************************************
                ****************************************************************
                **
                ** Name:       BDISPJ - HPIL Character-oriented display routine
                **
                ** Category:   PILI/O
                **
                ** Purpose:
                **      Routine to display characters on HPIL devices
                **
                ** Entry:
                **      A[B] is a data byte
                **      HEX mode
                **
                ** Exit:
                **      A[B] is the data byte from entry
                **      Display status bits restored
                **      HEX mode, carry clear
                **
                ** Calls:      CHKASN,SETLP,FNDMBX,START,GTYPE,MTYL,FINDA,
                **             GETMBX,WRITIT,SENDIT,SENDI+,PUTD,PUTX,END,
                **             MOVCUR,MOVCU+,DO=CUR,DO@CUR,Clear?,SendBf,
                **             BLANKC,LCleft,DSPCL?
                **
                ** Uses.......
                **  Exclusive: A[15:2],B[W],C[W],D[A],           D0,D1,P,(ST)
                **  Inclusive: A[15:2],B[W],C[W],D[15:13],D[5:0],D0,D1,P,(ST)
                **
                ** Stk lvls:   4 (START)
                **
                ** NOTE:
                **      Does not alter A[B], returns (DSPSTA+3) in STatus bits
                **
                ** History:
                **
                **    Date     Programmer            Modification
                **  --------   ----------   ---------------------------------
                **  09/28/83      NZ        Updated documentation
                **  06/24/83      NZ        Fixed bug of losing <Cr> if DISP
                **                          device is a printer device
                **  05/18/83      NZ        Changed return from GTYPE to
                **                          match new exit conditions of same
                **  04/14/83      NZ        Added check to ignore NULL char
```

```
                **  02/16/83      NZ          Removed Talker code (doesn't work
                **                            with multiple loop displays)
                **  12/09/82      NZ          Added documentation
                **
                ********************************************************************
                ********************************************************************
                 Esc    EQU    #1B            <Escape>
                 Bs     EQU    #08            <Backspace>
                *
                 RepCur EQU    0              Status bit...Replace the cursor
                *
                 Delete EQU    4              Status bit...Delete character
                 CurLft EQU    5              Status bit...Cursor direction
                 SetCur EQU    6              Status bit...Set vs move cursor
                 Protec EQU    SetCur         Status bit...Hit protected char?
                *
                *-
                *-
F3637           =BDISPJ
F3637 1B00      DISPoF  DO=(5) =IS-DSP        IS assignment
      000
F363E 15E6              C=DATO 7              Read it in...
F3642 7000              GOSUB  =CHKASN        Check if assigned...
F3646 560               GONC   DISP00         Assigned!
F3649 6683              GOTO   DISPOF         This is NOT assigned...return
                *-
                *-
F364D 1A00      DISP00  DO=(4) =DSPSET        Status nibble for display!
      00
F3653 0B                CSTEX
F3655 1562              C=DATO XS             Read in status...
F3659 0B                CSTEX
F365B 860               ?ST=0  =LoopOK
F365E BE                GOYES  DISPoF         Loop has been offed...exit now!
F3660 D7                D=C    A              Put address in D[A] for START
F3662 94E               ?C#0   S
F3665 32                GOYES  DISPNS         ...not current...set it up!
F3667 870               ?ST=1  =DispOK        Currently set up?
F366A F0                GOYES  DISP02         Yes...check if mailbox is there
                *
                * Display is NOT set up...check if this is a new assignment
                *
                * (New assignments have BOTH ST(=Wallby) and ST(=Printr) true)
                *
F366C 860               ?ST=0  =Wallby        Not HP82163A?
F366F C1                GOYES  DISPNO         No...this is NOT a new assignment
F3671 860               ?ST=0  =Printr        Not printer?
F3674 71                GOYES  DISPNO         No...this is NOT a new assignment
F3676 511               GONC   DISPNS         Go always...new assignment
                *-
                *-
                *
                * Now get back the correct loop for the display
                *
F3679 7000      DISP02  GOSUB  =SETLP         SETUP sets C[S] to current mbox
```

```
109 F367D 7000          GOSUB  =FNDMBX       FNDMBX sets MBOX^ to current mbox
110 F3681 460           GOC    DISPNS        If carry, not found...not set up?
111 F3684 68A0          GOTO   DISPOK
112               *-
113               *-
114 F3688 850   DISPNS  ST=1   =DispOK       Reuse this status as a flag!
115               *
116               * If ST(DispOK)=1, then need to check accessory ID here!
117               *
118 F368B       DISPNO
119               *
120               * Loop is NOT set up for DISPLAY IS!
121               *
122               * Save character on RSTK before calls to START, GTYPE, etc
123               *
124 F368B D6            C=A    A
125 F368D 06            RSTK=C                Push the character
126               *
127               * Call START, with device specifier in D[A]...
128               *
129 F368F 8E00          GOSUBL =START         Set up Loop
          00
130 F3695 424           GOC    DISPN.         Error
131 F3698 860           ?ST=0  =DispOK        Are the status bits OK already?
132 F369B 73            GOYES  DISPn4         Yes...continue!
133               *
134               * Get the accessory ID of the device in A[B]
135               *
136 F369D 8E00          GOSUBL =GTYPE         Returns Acc Id in A[B]
          00
137 F36A3 443           GOC    DISPN.         Error if carry
138               *
139               * If no response, then A[B] is zeroed by GTYPE
140               *
141               * Now set DSPSET true, set up other bits of DSPSET using B[B],
142               * then restore all and return
143               *
144 F36A6 840           ST=0   =Wallby        Preclear these statuses!
145 F36A9 840           ST=0   =Printr
146 F36AC 21            P=     1
147 F36AE 301           LCHEX  1              Mass storage class...
148 F36B1 902           ?A=C   P
149 F36B4 C4            GOYES  DISPN1         Error!!!
150               *
151 F36B6 302           LCHEX  2              Printer class...
152 F36B9 906           ?A#C   P              Is this a printer class device?
153 F36BC 80            GOYES  DISPn3         No...check if HP82163A
154               *
155               * Printer class device!
156               *
157 F36BE 850           ST=1   =Printr
158 F36C1 501           GONC   DISPn4         Go always!
159               *-
160               *-
161 F36C4 20    DISPn3  P=     0
```

```
162 F36C6 3103          LCHEX  30            HP82163A accessory id
163 F36CA 966           ?A#C   B
164 F36CD 50            GOYES  DISPn4        Not an HP82163A
165 F36CF 850           ST=1   =Wallby
166 F36D2       DISPn4
167             *
168             * Now set up the display as a listener (Acc ID in A[B])
169             *
170 F36D2 8E00          GOSUBL =MTYL         (Character is on RSTK)
          00
171 F36D8 472   DISPN.  GOC    DISPN1        Error
172 F36DB 850           ST=1   =DispOK       Display set up
173 F36DE 1A00          DO=(4) =DSPSET
          00
174 F36E4 0B            CSTEX
175 F36E6 1542          DATO=C XS            Write it back out
176 F36EA 0B            CSTEX
177 F36EC 1900          DO=(2) =IS-DSP
178 F36F0 DB            C=D    A
179 F36F2 15C2          DATO=C 3             Write out the address!
180 F36F6 07            C=RSTK
181 F36F8 DA            A=C    A             Restore the character to A[B]
182 F36FA 20            P=     0
183 F36FC 6030          GOTO   DISPOK
184             *-
185             *-
186             *
187             * If here, had a loop error...clear DISPLAY IS
188             *
189 F3700 07    DISPN1  C=RSTK
190 F3702 DA            A=C    A             Restore character from RSTK
191 F3704 1B00          DO=(5) (=IS-DSP)+3   Status nibble...
          000
192 F370B 1562          C=DATO XS
193 F370F 0B            CSTEX
194 F3711 85B           ST=1   11            Set "OFF"ed flag
195 F3714 0B            CSTEX
196 F3716 F2            CSL    A             Move to C[3]
197 F3718 182           DO=DO- 3             Point to IS-DSP
198 F371B AB2           C=0    X
199 F371E A3E           C=C-1  X             C[X]=FFF
200 F3721 15C3          DATO=C 4             OFF the display
201 F3725 6552          GOTO   DISPEX        Done!
202             *-
203             *-
204 F3729 60A2  DISPOx  GOTO   DISPOX        Done, don't check carry
205             *-
206             *-
207             *
208             * Loop is set up now!
209             *
210 F372D       DISPOK
211             *
212             * First ensure that not in an escape sequence!!!!
213             *
```

```
    214 F372D 1B00          D0=(5) =ESCSTR       Escape status
              000
    215 F3734 15E0          C=DAT0 1             Read it...
    216 F3738 A0E           C=C-1  P             ...decrement it...
    217 F373B 4B4           GOC    DISPnE        Not escape
    218               *
    219               * This is in an escape sequence...what do I do?
    220               *
    221               *
    222               * Check if printer...if so, return
    223               *
    224 F373E 870           ?ST=1  =Printr
    225 F3741 8E            GOYES  DISP0x        Exit, restore all levels
    226               *
    227               * Not a printer...continue
    228               *
    229 F3743 90A           ?C=0   P             Is it "escape"?
    230 F3746 90            GOYES  DISP1         Yes...check further
    231 F3748 846  DspsnO   ST=0   SetCur        No...send the character without
    232 F374B 6E52          GOTO   DspSnO          repositioning the cursor
    233               *
    234               * Escape mode
    235               *
    236 F374F 844   DISP1   ST=0   Delete        Assume NOT a delete until proven
    237               *                          otherwise!!!
    238 F3752 8F00          GOSBVL =FINDA        A[B] is value
              000
    239 F3759 34            CON(2) \C\           Right arrow
    240 F375B 6C0           REL(3) RArrow
    241 F375E 44            CON(2) \D\           Left arrow
    242 F3760 BD0           REL(3) LArrow
    243 F3763 05            CON(2) \P\           Delete character
    244 F3765 890           REL(3) DelChr
    245 F3768 F4            CON(2) \O\           Delete character with wrap
    246 F376A 390           REL(3) DelChr
    247 F376D E4            CON(2) \N\           Insert char with wrap
    248 F376F 990           REL(3) InsChr
    249 F3772 B4            CON(2) \K\           Delete through end of line
    250 F3774 211           REL(3) DelLin
    251 F3777 30            CON(2) 3             Cursor far right
    252 F3779 8C0           REL(3) FarRt
    253 F377C 40            CON(2) 4             Cursor far left
    254 F377E 201           REL(3) FarLft
    255 F3781 00            CON(2) 0             Others...
    256 F3783 6DF1          GOTO   EscSnd        Send <Esc> <character> & return
    257               *-
    258               *-
    259               *
    260               * If <Lf>: Send it immediately, independent of current mode
    261               * If <Cr>: If (not Printr): send immediately (Don't set cursor)
    262               *                     else: transmit buffer, then <Cr>
    263               * If chr$(0): Ignore it entirely if not in escape sequence
    264               * If <anything else> and <Printr>: return without action
    265               *
    266 F3787 968   DISPnE  ?A=0   B             Is A[B]=0?
```

```
267 F378A F9          GOYES  DISPOx        Yes...exit.
268 F378C 31A0        LCHEX  0A            <Lf>
269 F3790 962         ?A=C   B
270 F3793 5B          GOYES  DspsnO        Send it!
271 F3795 30D         LCHEX  D             Preload <Cr>
272 F3798 870         ?ST=1  =Printr
273 F379B B0          GOYES  DISP.1        Check further in printer code
274 F379D 962         ?A=C   B             Is it a <Cr>?
275 F37A0 8A          GOYES  DspsnO        Yes...don't reposition the cursor
276 F37A2 6051        GOTO   DISP2         No...process the character
277             *-
278             *-
279 F37A6 966 DISP.1  ?A#C   B             Is it a <Cr>?
280 F37A9 08          GOYES  DISPOx        No...Exit, no action!
281 F37AB 7BF3        GOSUB  Clear?        Is the clear flag set?
282 F37AF 489         GOC    DspsnO        Yes...send only the <Cr>
283             *
284             * This is a printer, and I got a <Cr>...
285             * need to send whole buffer!!!
286             *
287 F37B2 1900        DO=(2) (=DSPBFS)-2   (Clear? leaves DO @ DSPSTA+3)
288 F37B6 31F5        LC(2)  95
289 F37BA 161 DISP.2  DO=DO+ 2
290 F37BD 14A         A=DATO B
291 F37C0 968         ?A=0   B
292 F37C3 80          GOYES  DISP.3        End of buffer (Logical)!
293 F37C5 A6E         C=C-1  B             End of buffer (Physical)?
294 F37C8 51F         GONC   DISP.2        No...try next character
295             *
296             * Now DO points to first "non-character"
297             *
298 F37CB AF0 DISP.3  A=0    W             Clear for ASRB below
299 F37CE 132         ADOEX
300 F37D1 3400        LC(5)  =DSPBFS
          000
301 F37D8 135         D1=C                 Set D1 @ DSPBFS also
302 F37DB EA          A=A-C  A
303 F37DD 81C         ASRB                 Now A[A] is # of characters
304             *
305             * Set up for HPIL transfer
306             *
307 F37E0 7000        GOSUB  =GETMBX       Restore the HPIL mailbox to DO
308 F37E4 8E00        GOSUBL =WRITIT       Send the buffer
          00
309 F37EA 20          P=     0
310 F37EC 31D0        LCHEX  0D            Restore the <Cr>
311 F37F0 DA          A=C    A
312 F37F2 460         GOC    DISPEx        Exit if error!
313 F37F5 7BC3        GOSUB  Putd          Send it to the printer
314 F37F9 6181 DISPEx GOTO   DISPEX
315             *-
316             *-
317             *
318             * Code to check if Insert or Delete!
319             *
```

```
320 F37FD       DelChr
321             *
322             * Delete character (Either HP82163A or "other")
323             *
324 F37FD 854           ST=1    Delete      This IS a delete!
325 F3800 7BD1          GOSUB   SendBf      Send to end of line
326 F3804 6671          GOTO    DISPEX      Restore, etc.
327             *-
328             *-
329 F3808       InsChr
330             *
331             * Insert character (Sequence to turn on mode)
332             *
333 F3808 7000          GOSUB   =GETMBX     Get back the mailbox first
334 F380C 35B1          LCHEX   1B511B      Esc Q Esc
          15B1
335 F3814 8E00          GOSUBL  =PUTX
          00
336 F381A 4ED           GOC     DISPEx      Error if carry
337 F381D 6A2F          GOTO    Dspsn0      Now send the current char
338             *-
339             *-
340 F3821       RArrow
341             *
342             * Right arrow!
343             *
344 F3821 75A3          GOSUB   D0@CUR
345 F3825 14E           C=DAT0  B
346 F3828 96A           ?C=0    B
347 F382B C0            GOYES   DISPox      At end of buffer NOW!
348 F382D 845           ST=0    CurLft
349 F3830 846  Arrow    ST=0    SetCur      This is NOT just a set, but MOVE!
350 F3833 71C2          GOSUB   MOVCUR
351 F3837 6291 DISPox   GOTO    DISPOX      Not interrupted (For sure!)
352             *-
353             *-
354 F383B       LArrow
355             *
356             * Left arrow!
357             *
358 F383B 855           ST=1    CurLft
359 F383E 51F           GONC    Arrow       Go always! (FINDA:RTNCC)
360             *-
361             *-
362 F3841       FarRt
363             *
364             * Cursor far right!
365             *
366 F3841 845           ST=0    CurLft      This is cursor RIGHT
367 F3844 7283 Farxx    GOSUB   D0@CUR      C[B] is current cursor value
368 F3848 DA            A=C     A           Save cursor value in A[B]
369 F384A 846           ST=0    SetCur      This is NOT just a SET, but MOVE!
370 F384D 875  FarRt1   ?ST=1   CurLft      Is this LEFT?
371 F3850 E0            GOYES   FarRt2      Yes...don't check for end!
372 F3852 7473          GOSUB   D0@CUR      No...check if at end already
```

```
373 F3856 14E          C=DATO B
374 F3859 96A          ?C=0   B
375 F385C C0           GOYES  FarRt3        Already at far right of buffer!
376 F385E 850  FarRt2  ST=1   RepCur        Reposition the cursor at new loc.
377 F3861 7692         GOSUB  MOVCU+
378 F3865 57E          GONC   FarRt1        Moved it...move it again!
379 F3868 3130 FarRt3  LC(2)  3             Cursor far right!
380 F386C 865          ?ST=0  CurLft        Is this RIGHT?
381 F386F 40           GOYES  FarEnd        Yes...exit!
382 F3871 E6           C=C+1  A             No...(LEFT=4)
383 F3873 7B63 FarEnd  GOSUB  D0=CUR
384 F3877 148          DATO=A B             Restore the cursor value
385 F387A DA           A=C    A
386 F387C 6D41         GOTO   DISPOX        Finish it up!
387             *-
388             *-
389 F3880       FarLft
390 F3880 855          ST=1   CurLft
391 F3883 50C          GONC   Farxx         Go always!
392             *-
393             *-
394             *
395             * Delete through end of line
396             *
397 F3886 8F00 DelLin  GOSBVL =SCNRT
          000
398             *
399             * Check if no protected fields after this...if none, send
400             * <Esc> J (Clear to end of screen)
401             *
402 F388D 4D0          GOC    DelL0         If carry, reached end of buffer!
403 F3890 130          D0=A
404 F3893 14E          C=DATO B             Read in indicated character
405 F3896 96E          ?C#0   B
406 F3899 B1           GOYES  DelL1         Protected field!
407 F389B 7000 DelL0   GOSUB  =GETMBX
408 F389F 7D13         GOSUB  PUTEsc        Send Esc
409 F38A3 4C0          GOC    Delexc        Carry exit
410 F38A6 31A4         LCASC  \J\
411 F38AA 7613         GOSUB  Putd          Send the "J"
412             *
413             * Following two lines can be packed to one by GOTO DelEx
414             *
415 F38AE D4           A=B    A             Copy the "K" back!
416 F38B0 6AC0 Delexc  GOTO   DISPEX
417             *-
418             *-
419             *
420             * Delete to protected field!
421             *
422 F38B4 AF2  DelL1   C=0    W
423 F38B7 DB           C=D    A
424 F38B9 EE           C=A-C  A
425 F38BB 81E          CSRB
426 F38BE E6           C=C+1  A             Increment for current character
```

```
427                *
428                * Now C[A] is count of blanks to send
429                *
430 F38C0 DA           A=C    A             Copy count to A[A]
431 F38C2 D7           D=C    A             D[A]=count
432 F38C4 8E00         GOSUBL =BLANKC       Blanks (Clear the items)
          00
433 F38CA AF5          B=C    W             Copy to B[7:0]
434                *
435                * This will NOT work for a non-HP82163A device in INSERT mode
436                * (Will insert n blanks, where n is the # characters to the
437                * start of the protected field)
438                *
439 F38CD 8E00         GOSUBL =SENDI+       Get mailbox, Send A[A] blanks
          00
440 F38D3 451          GOC    DelEx         If carry, abort!
441                *
442                * Now back up to starting point
443                *
444 F38D6 DB           C=D    A
445 F38D8 DA           A=C    A             Count to A[A]
446 F38DA 7D03         GOSUB  LCleft        (Loads C with <Esc> D <Esc> D)
447 F38DE AF5          B=C    W
448 F38E1 C4           A=A+A  A             Double count for <Esc> D
449 F38E3 8E00         GOSUBL =SENDIT       Send backspaces
          00
450 F38E9 31B4 DelEx   LCASC  \K\           Restore original character (K)
451 F38ED DA           A=C    A
452 F38EF 6B80         GOTO   DISPEX        Done...exit
453                *-
454                *-
455 F38F3      DISP2
456                *
457                * Check if it is an <Esc>...if so, do NOTHING until next char
458                *
459 F38F3 31B1         LC(2)  Esc
460 F38F7 962          ?A=C   B             Is this an escape?
461 F38FA F0           GOYES  DISPoX        Yes...exit, no change
462                *
463                * Check if backspace - if so, do a backspace and return
464                *
465 F38FC 3180         LC(2)  Bs            <Bs>
466 F3900 966          ?A#C   B             Is this a backspace?
467 F3903 A0           GOYES  DISP25        No...check further
468                *
469                * This is a backspace!
470                *
471 F3905 653F         GOTO   LArrow        Carry MUST be clear for LArrow
472                *-
473                *-
474 F3909 60C0 DISPoX  GOTO   DISPOX        Jump (GOYES out of range)
475                *-
476                *-
477 F390D 1B00 DISP25  DO=(5) =DSPSTA
          000
```

```
478 F3914 15E2          C=DATO 3
479 F3918 0A            ST=C                    Restore user status for DSPCL?
480 F391A 8F00          GOSBVL =DSPCL?
          000
481 F3921 1A00          D0=(4) (=DSPSTA)+3      Restore display status for me
          00
482 F3927 15E2          C=DATO 3
483 F392B 1A00          D0=(4) =DSPSET          Point to the HPIL status nibble
          00
484 F3931 1562          C=DATO XS               Recall the HPIL status from RAM
485 F3935 0A            ST=C
486               *
487               * Check if cursor is at end of buffer!
488               *
489 F3937 1A00          D0=(4) =CURSOR
          00
490 F393D 14E           C=DATO B
491 F3940 D5            B=C    A                Copy cursor value to B[B]
492 F3942 31F5          LC(2)  95
493 F3946 9E5           ?B<C   B                Reached physical end of buffer?
494 F3949 02            GOYES  DISP30           No...check if insert mode
495               *
496               * Cursor is at end of buffer...check if insert or replace mode
497               *
498 F394B 870           ?ST=1  =Insert
499 F394E BB            GOYES  DISPoX           Exit, no error (no room)
500               *
501               * At end of buffer, not insert...send char, backspace
502               *
503 F3950 7000          GOSUB  =GETMBX          Get mailbox
504 F3954 3500          LCHEX  441B00
          B144
505 F395C AE6           C=A    B                (char)&<esc>&"D"
506 F395F 8E00          GOSUBL =PUTX            Send it!
          00
507 F3965 6510          GOTO   DISPEX           Exit
508               *-
509               *-
510 F3969       DISP30
511               *
512               * Cursor is NOT at end of buffer...check if insert or replace
513               *
514 F3969 860           ?ST=0  =Insert          Insert mode?
515 F396C B3            GOYES  DspSnd           Not Insert...send the char!
516               *
517               * Insert mode...call SendBf (It checks for HP82163A)
518               *
519 F396E 844           ST=0   Delete           This is NOT delete!
520 F3971 7A60          GOSUB  SendBf           Send to end of line
521 F3975 856           ST=1   SetCur           Set the cursor to new spot...
522 F3978 534           GONC   DspSn2           If OK, position it!
523               *
524               * Following jump taken ONLY if entered through DISPEX
525               * (Packing technique)
526               *
```

```
527 F397B 5E4  DISPEX  GONC   DISPOX        If no carry, finish up
528 F397E 401          GOC    DspErr        Go always!
529            *-
530            *-
531 F3981 7000 EscSnd  GOSUB  =GETMBX       Get the mailbox first...
532 F3985 7732         GOSUB  PUTEsc        ...Send the <Esc>...
533 F3989 846          ST=0   SetCur        ...DON'T set the cursor!
534 F398C 512          GONC   DspSn1        Go unless interrupted
535 F398F 840  DspErr  ST=0   =LoopOK       Interrupted!
536 F3992 840          ST=0   =DispOK       (If interrupted, display not OK)
537 F3995 1B00         DO=(5) =DSPSET       Rewrite display settings!
          000
538 F399C 0B           CSTEX
539 F399E 1542         DATO=C XS
540 F39A2 0B           CSTEX
541 F39A4 452          GOC    DISPOX        Go always...exit
542            *-
543            *-
544 F39A7      DspSnd
545            *
546            * Send the character and return
547            *
548 F39A7 856          ST=1   SetCur        SET the cursor to next position
549 F39AA 7000 DspSn0  GOSUB  =GETMBX       Find the mailbox...
550 F39AE D6   DspSn1  C=A    A             ...copy character to C[B]...
551 F39B0 7012         GOSUB  Putd          ...Send the character
552 F39B4 4AD          GOC    DspErr        Interrupted!
553 F39B7 866          ?ST=0  SetCur        Set the new cursor position?
554 F39BA 01           GOYES  DISPOX        No...exit
555 F39BC 7AE1 DspSn2  GOSUB  Clear?        Check if Clear is set
556 F39C0 490          GOC    DISPOX        Yes...exit (Don't move cursor)
557 F39C3 845          ST=0   CurLft        No...move the cursor RIGHT
558 F39C6 7E21         GOSUB  MOVCUR
559 F39CA      DISPOX
560            *
561            * Following line is not needed anymore, but is a residual from
562            * earlier code (could be removed)
563            *
564 F39CA 8E00         GOSUBL =END          Clean up the loop(A[A] unchanged)
          00
565            *
566            * Now restore status bits and return
567            *
568 F39D0      DISPOF
569 F39D0 1B00         DO=(5) (=DSPSTA)+3   Display status bits
          000
570 F39D7 15E2         C=DATO 3
571 F39DB 0A           ST=C                 Restore them!
572 F39DD 03           RTNCC                Done...return, carry clear
573            ********************************************************************
574            ********************************************************************
575            **
576            ** Name:       SendBf - Insert/delete a char, send line if needed
577            **
578            ** Category:   LOCAL
```

```
                **
                ** Purpose:
                **      Insert/delete a character, even if this is an HP82163A
                **      display device
                **
                ** Entry:
                **      ST(Insert):
                **              if 1, insert (send from position through end)
                **                 (send character from A[B] first!)
                **      ST(Delete) is type:
                **              if 1, delete (send from next char to end,
                **                 append blank)
                **              if 0, insert (send char from A[B], then to end)
                **
                ** Exit:
                **      A[B] is not changed from entry
                **      P=0
                **      Carry set if interrupted, clear if OK
                **
                ** Calls:       SCNRT,GETMBX,PUTEsc,PUTD,WRITIT,LCleft
                **
                ** Uses.......
                **  Exclusive: A[15:2],B[W],C[W],     D0,D1   ST[Protec]
                **  Inclusive: A[15:2],B[W],C[W],D[A],D0,D1,P,ST[Protec,3:0]
                **
                ** Stk lvls:   2 (WRITIT)(SCNRT)
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **   --------   ----------    --------------------------------
                **  06/24/83       NZ         Packed code by no longer preserve
                **                            D1 in this routine
                **  06/02/83       NZ         Added code to do Esc N (Insert w/
                **                            wrap)
                **  12/09/82       NZ         Added documentation
                **
                ********************************************************************
                ********************************************************************
F39DF           =SendBf
                *
                * Find first character NOT to send (Either EOB or protected)
                *
F39DF 8F00          GOSBVL =SCNRT       Scan right
      000
                *
                * SCNRT returns A[A]-->past unprotected item, carry set if end
                * of buffer, D[A] is pointer to first after current position,
                * B[B] contains the entry A[B]
                *
F39E6 5C0           GONC   NotEnd       If carry, at end of buffer
                *
                * If Insert and End of buffer, return (Do nothing)
                *
F39E9 860           ?ST=0  =Insert      Is it NOT insert?
```

```
F39EC 70          GOYES  NotEnd       Not insert...continue
F39EE 864         ?ST=0  Delete       Is it a delete?
F39F1 A4          GOYES  Sendex       No...buffer is full, insert: exit
F39F3     NotEnd
          *
          * B[B] is the new character...saved here for now
          *
          * D[A] is first char after current position in buffer
          *
F39F3 AF2         C=0    W            Clear high bits for CSRB below
F39F6 DB          C=D    A            Start of string in C[A]
F39F8 135         D1=C                Start of string in D1
F39FB 846         ST=0   Protec       Check if protected field
F39FE 130         D0=A
F3A01 14E         C=DAT0 B
F3A04 96A         ?C=0   B
F3A07 50          GOYES  NotPro       Not protected (EOB)
F3A09 856         ST=1   Protec
F3A0C 137 NotPro  CD1EX               Bring pointer back to C[A]...
F3A0F 135         D1=C                ...And copy back to D1
F3A12 EE          C=A-C  A            # of nibbles to send
F3A14 81E         CSRB                C[A] is length to send (bytes)
F3A17 DA          A=C    A            A[A] is length to send (bytes)
          *
          * Now D1 points past start of buffer, A[A] is a character count
          *
          * Get the mailbox address into D0 now...
          *
F3A19 7000        GOSUB  =GETMBX      Alters only C,D0
          *
          * Now D0 points to the mailbox
          *
          * Check if Protec is set...if so, and in insert mode, and not
          * HP82163A, then send <Esc>R to turn OFF insert mode
          *
F3A1D 870         ?ST=1  =Wallby      HP82163A?
F3A20 63          GOYES  Send#        Yes...continue
F3A22 876         ?ST=1  Protec       Protected?
F3A25 A1          GOYES  Send-        Yes...continue
          *
          * Not HP82163A, not protected...just send the char (or delete
          * escape sequence)
          *
F3A27 D0          A=0    A
F3A29 864         ?ST=0  Delete       Is this a delete?
F3A2C 90          GOYES  Send+        No...just the character
F3A2E 7E81        GOSUB  PUTEsc       Yes...send Esc...
F3A32 480         GOC    Sendex
F3A35 D9  Send+   C=B    A            Copy B[B] (the character)
F3A37 7981        GOSUB  Putd         ...send the character
F3A3B D4  Sendex  A=B    A            Restore the character from B[B]
F3A3D 03          RTNCC               Exit!
          *-
          *-
          *
```

```
688             * This is not HP82163A, Protected!
689             *
690 F3A3F 860    Send-   ?ST=0  =Insert       Insert mode?
691 F3A42 41            GOYES  Send#          No...continue
692             *
693             * This is not HP82163A, protected, insert mode...temporarily
694             * disable insert mode
695             *
696 F3A44 7871          GOSUB  PUTEsc         PUT Esc...
697 F3A48 42F           GOC    Sendex
698 F3A4B 3125          LCASC  \R\            ...R
699 F3A4F 7171          GOSUB  Putd
700 F3A53 47E           GOC    Sendex         Error...restore, return
701             *
702             * Check if insert...if so, send the character in B[B] first!
703             * If not insert (if delete), skip first character in buffer
704             *
705             * Check if this is the logical end of buffer
706             *
707 F3A56 1C1   Send#   D1=D1- 2              Point to first character
708 F3A59 14F           C=DAT1 B              Check the character for EOB
709 F3A5C 171           D1=D1+ 2              Restore pointer to next char
710 F3A5F 96E           ?C#0   B              End of line?
711 F3A62 40            GOYES  Send00         No...check if need to adjust
712 F3A64 D0            A=0    A              Yes...set =0
713 F3A66 874   Send00  ?ST=1  Delete         Delete?
714 F3A69 A1            GOYES  SendNI         Yes...NOT insert!
715             *
716             * This is an insert!
717             *
718 F3A6B 1C1           D1=D1- 2              This is an insert...
719 F3A6E 96A           ?C=0   B              Is it End of buffer?
720 F3A71 90            GOYES  Send02         Yes...skip this adjustment
721 F3A73 876           ?ST=1  Protec         Is it protected?
722 F3A76 40            GOYES  Send02         Yes...leave A[A] unchanged
723 F3A78 E4            A=A+1  A              Increment count
724             *
725             * Now A[A] is corrected character count, D1 @ first char to
726             * be sent
727             *
728 F3A7A D9    Send02  C=B    A              Read the character from B[B]
729 F3A7C 7441          GOSUB  Putd           Send the character
730 F3A80 4AB           GOC    Sendex         Error...exit
731             *
732             * This is the entry point for a delete!
733             *
734 F3A83       SendNI
735             *
736             * Now retransmit the line...
737             *
738 F3A83 8E00          GOSUBL =WRITIT        Send the data to the loop
          00
739             *
740             * If carry set, ATTN hit...return
741             *
```

```
742 F3A89 41B          GOC    Sendex        Exit after restoring A[B]
743             *
744             * Done with transfer now...check if delete; if so, send blank
745             *
746 F3A8C 864          ?ST=0  Delete
747 F3A8F D0           GOYES  SendLs        Insert...no trailing blank
748 F3A91 3102         LCASC  \ \           Delete...
749 F3A95 7B21         GOSUB  Putd          ...send a trailing blank
750 F3A99 41A          GOC    Sendex        Exit if error
751             *
752             * Now D1 points to the "Next" character...subtract current
753             * position and divide by 2 to get # of bytes sent
754             *
755 F3A9C 866  SendLs   ?ST=0  Protec        Is this NOT protected field?
756 F3A9F 81           GOYES  SendL1        Not protected...back up
757 F3AA1 870          ?ST=1  =Wallby       Is this an HP82163A?
758 F3AA4 31           GOYES  SendL1        Yes...back up
759 F3AA6 860          ?ST=0  =Insert       Am I in insert mode?
760 F3AA9 E0           GOYES  SendL1        No...back up
761 F3AAB 7111         GOSUB  PUTEsc        Send <Esc>N to turn insert on
762 F3AAF 31E4         LCASC  \N\
763 F3AB3 7D01         GOSUB  Putd
764 F3AB7 AF0  SendL1  A=0    W             Clear high bits for ASRB below
765 F3ABA 133          AD1EX
766 F3ABD 3400         LC(5)  =DSPBFS
          000
767 F3AC4 EA           A=A-C  A
768 F3AC6 81C          ASRB                 A[A] is # bytes from buffer start
769 F3AC9 1F00         D1=(5) =CURSOR
          000
770 F3AD0 D2           C=0    A
771 F3AD2 14F          C=DAT1 B             Read the cursor...
772 F3AD5 EA           A=A-C  A             Now A[A] is # backspaces to send
773 F3AD7 C4           A=A+A  A             Double for <Esc> D
774 F3AD9 DC           ABEX   A             Now character in A[B], # in B[A]
775 F3ADB 814          ASRC
776 F3ADE 814          ASRC                 Save character in A[15:14]
777 F3AE1 D4           A=B    A             Count back to A[A]
778 F3AE3 7401         GOSUB  LCleft        Load C with Esc D Esc D
779 F3AE7 AF5          B=C    W
780 F3AEA 8E00         GOSUBL =SENDIT       Send the sequence!
          00
781 F3AF0 810          ASLC
782 F3AF3 810          ASLC                 Restore A[B] from A[15:14]
783 F3AF6 01           RTN                  Don't alter carry!
784             ****************************************************************
785             ****************************************************************
786             **
787             ** Name:       MOVCUR - Move the cursor right/left
788             ** Name:       MOVCU+ - Move the cursor permanently (no restore)
789             **
790             ** Category:   LOCAL
791             **
792             ** Purpose:
793             **      Move the cursor in the direction specified by CurLft
```

```
                **      status bit (Similar to mainframe routine by same name)
                **
                ** Entry:
                **      CurLft set to move left, clear to move right
                **      P=0
                **
                ** Exit:
                **      Contents of A[A] restored upon exit
                **      Carry set if no move
                **      Carry clear if moved, cursor positioned on display
                **      Clears ST(=LoopOK) if interrupted
                **      P=0
                **
                ** Calls:      DO=CUR,MOVC60,GETMSK,SENDI+,LCleft
                **
                ** Uses.......
                **  Exclusive: A[15:5],B[W],C[W],D[A],   P
                **  Inclusive: A[15:5],B[W],C[W],D[A],DO,P,ST[3:0]
                **
                ** Stk lvls:   2 (SENDI+)
                **
                ** NOTE: Does not alter A[A]
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------   ---------------------------------
                **  12/09/82      NZ         Added documentation
                **
                ****************************************************************
                ****************************************************************
F3AF8 840    MOVCUR ST=0   RepCur        Do NOT replace cursor!
F3AFB 73E0   MOVCU+ GOSUB  DO=CUR
F3AFF 14E           C=DATO B
F3B02 D7            D=C    A            Save original value in D[B]
F3B04 D8            B=A    A            Save original character in B[A]
F3B06 14A    MOVC10 A=DATO B
F3B09 7090          GOSUB  MOVC60
F3B0D 31F5          LC(2)  95
F3B11 9E6           ?A>C   B            Would this be past end of display?
F3B14 C6            GOYES  MOVC50       Yes, then restore original value
F3B16 148           DATO=A B            No, then update cursor position
F3B19 D4            A=B    A            Save original char in A[B]
F3B1B 8F00          GOSBVL =GETMSK      Get bit map (Alters B[A],C,DO,P)
      000
F3B22 D8            B=A    A            Resave original char in B[B]
F3B24 15A0          A=DATO 1            Read mask nibble
F3B28 0E06          A=A&C  P
F3B2C 72B0          GOSUB  DO=CUR
F3B30 90C           ?A#0   P            Is it protected?
F3B33 3D            GOYES  MOVC10       Yes, then keep looking
                *
                * Now calculate how far to move cursor, and which direction...
                * ...and restore cursor value!!!
                *
```

```
 848 F3B35 D0          A=0    A            Clear high nibbles of A[A]
 849 F3B37 14A         A=DATO B            Read in cursor position
 850 F3B3A DB          C=D    A
 851 F3B3C 870         ?ST=1  RepCur       Replace the cursor?
 852 F3B3F 50          GOYES  MOVC15       Yes...don't restore it!
 853 F3B41 14C         DATO=C B            Restore original cursor position
 854 F3B44 B6A  MOVC15 A=A-C  B            Offset (Bytes) in A[B]
 855 F3B47 37B1        LCHEX  431B431B     Right arrows!
           34B1
           34
 856 F3B51 590         GONC   MOVC20       If carry, left arrow!
 857               *
 858               * Left arrows needed!
 859               *
 860 F3B54 7390 MOVC17 GOSUB  LCleft       Left arrows!
 861 F3B58 BE8         A=-A   B
 862 F3B5B AFD  MOVC20 BCEX   W            Move arrows to B[W], char to C[B]
 863 F3B5E D7          D=C    A            Save char in D[B]
 864 F3B60 866         ?ST=0  SetCur       Is this a move or a set?
 865 F3B63 90          GOYES  MOVC30       No...MOVE that # of chars
 866               *
 867               * This is a set cursor...if next char is the destination, exit
 868               *
 869 F3B65 CC          A=A-1  A
 870 F3B67 8A8         ?A=0   A
 871 F3B6A 21          GOYES  MOVC45       Exit w/o sending any(char in C,D)
 872               *
 873               * Must MOVE the cursor...send <Esc> C|D (A is # moves)
 874               *
 875 F3B6C C4   MOVC30 A=A+A  A            Double for <Esc>
 876 F3B6E 8E00        GOSUBL =SENDI+      Get mailbox, send left arrows
           00
 877 F3B74 550         GONC   MOVC40       No interrupt...ok
 878 F3B77 840         ST=0   =LoopOK      Interrupt...clear =LoopOK
 879 F3B7A DB   MOVC40 C=D    A
 880 F3B7C DA   MOVC45 A=C    A            Restore the original character...
 881 F3B7E 03          RTNCC               ...and return!
 882               *-
 883               *-
 884 F3B80      MOVC50
 885 F3B80 866         ?ST=0  SetCur       Is it NOT SetCur?
 886 F3B83 11          GOYES  MOVC55       Not SetCur...OK to not move
 887               *
 888               * SetCur...need to take action if unable to move right!
 889               *
 890               * First restore the cursor
 891               *
 892 F3B85 DB          C=D    A
 893 F3B87 14C         DATO=C B            DO is still at cursor...
 894 F3B8A D0          A=0    A
 895 F3B8C A6C         A=A-1  B
 896 F3B8F CC          A=A-1  A            A[B]=FE (A=-A B will make this 2)
 897               *
 898               * Go move the cursor left 1 position (since this is SetCur,
 899               * MOVC17 reduces the count by one, therefore A[B] is now -2)
```

```
900                *
901 F3B91 52C              GONC   MOVC17        Go always
902                *-
903                *-
904 F3B94           MOVC55
905 F3B94 DB               C=D    A             C(B)=Original cursor
906 F3B96 14C              DAT0=C B             Restore original cursor
907 F3B99 D4               A=B    A             Restore original char from B[B]
908 F3B9B 02               RTNSC
909                *-
910                *-
911 F3B9D 865       MOVC60 ?ST=0  CurLft        Moving cursor left?
912 F3BA0 60               GOYES  MOVC70        No, then skip
913 F3BA2 CC               A=A-1  A             Yes, then decrement value
914 F3BA4 01               RTN
915                *-
916                *-
917 F3BA6 E4        MOVC70 A=A+1  A             Increment value
918 F3BA8 01               RTN
919                *-
920                *-
921                ********************************************************************
922                ********************************************************************
923                **
924                ** Name:       Clear? - Check if the clear bit is set in DSPSTA
925                **
926                ** Category:   LOCAL
927                **
928                ** Purpose:
929                **      Set/clear carry if clear bit in DSPSTA is set/clear
930                **
931                ** Entry:
932                **      None
933                **
934                ** Exit:
935                **      Carry set if ST[Clear] is set, else clear
936                **      D0 @ DSPSTA+3
937                **
938                ** Calls:      None
939                **
940                ** Uses.......
941                **  Inclusive: C[X],D0
942                **
943                ** Stk lvls:   0
944                **
945                ** History:
946                **
947                **    Date     Programmer             Modification
948                **  --------   ----------   ---------------------------------
949                **  09/28/83      NZ        Added documentation
950                **
951                ********************************************************************
952                ********************************************************************
953 F3BAA 1B00 =Clear? D0=(5) (=DSPSTA)+3   Point to status
          000
```

```
954 F3BB1 15E2          C=DAT0 3            Read in 3 nibbles of status
955 F3BB5 0B            CSTEX               Now check if CLEAR is set...
956 F3BB7 870           ?ST=1  =Clear
957 F3BBA 20            GOYES  Clear1       Set/clear carry...
958 F3BBC 0B    Clear1  CSTEX               (Restore my status)
959             *
960             * If carry set, then =Clear is set
961             *
962 F3BBE 01            RTN                 Return, carry unchanged
963             *-
964             *-
965 F3BC0 31B1 =PUTEsc LC(2)  Esc
966 F3BC4 8C00 Putd    GOLONG =PUTD
          00
967             ********************************************************************
968             ********************************************************************
969             **
970             ** Name:       D0@CUR - Set D0 to the current cursor position
971             **
972             ** Category:   LOCAL
973             **
974             ** Purpose:
975             **      Set D0 to the cursor position in the display
976             **
977             ** Entry:
978             **      None
979             **
980             ** Exit:
981             **      D0 at cursor position
982             **      Carry clear
983             **      C[A] is cursor value (from =CURSOR)
984             **
985             ** Calls:      D0=CUR
986             **
987             ** Uses.......
988             **  Inclusive: C[A],D0
989             **
990             ** Stk lvls:   1 (D0=CUR)
991             **
992             ** History:
993             **
994             **    Date      Programmer          Modification
995             **  --------    ----------    --------------------------------
996             **  02/18/83      NZ          Added D0=CUR call, renamed to
997             **                            D0@CUR
998             **  12/09/82      NZ          Added documentation
999             **
1000            ********************************************************************
1001            ********************************************************************
1002 F3BCA 7410 =D0@CUR GOSUB  D0=CUR       Leaves D0 pointing to cursor loc
1003 F3BCE D2           C=0    A
1004 F3BD0 14E          C=DAT0 B
1005 F3BD3 161          D0=D0+ 2            (=CURSOR)-(=DSPBFS)
1006 F3BD6 132          ADOEX               Save A[A] in D0, set A[A] to DSPBFS
1007 F3BD9 CA           A=C+A  A
```

```
1008 F3BDB CA           A=C+A  A
1009 F3BDD 132          ADOEX                 Restore A[A], set DO to cursor
1010 F3BE0 03           RTNCC
1011            *-
1012            *-
1013 F3BE2 1B00  DO=CUR DO=(5) =CURSOR
           000
1014 F3BE9 01           RTN
1015            *-
1016            *-
1017 F3BEB 37B1  LCleft LCHEX  441B441B       Esc D Esc D
           44B1
           44
1018 F3BF5 01           RTN
1019 F3BF7              END
```

```
 Arrow   Abs  997424 #F3830 -   349   359
=BDISPJ  Abs  996919 #F3637 -    74
 BLANKC  Ext                -   432
 Bs      Abs       8 #00008 -    63   465
 CHKASN  Ext                -    77
 CURSOR  Ext                -   489   769  1013
 Clear   Ext                -   956
 Clear1  Abs  998332 #F3BBC -   958   957
=Clear?  Abs  998314 #F3BAA -   953   281   555
 CurLft  Abs       5 #00005 -    68   348   358   366   370   380   390   557
                              911
 D0=CUR  Abs  998370 #F3BE2 -  1013   383   826   841  1002
=D0@CUR  Abs  998346 #F3BCA -  1002   344   367   372
 DISP.1  Abs  997286 #F37A6 -   279   273
 DISP.2  Abs  997306 #F37BA -   289   294
 DISP.3  Abs  997323 #F37CB -   298   292
 DISP00  Abs  996941 #F364D -    82    78
 DISP02  Abs  996985 #F3679 -   108    92
 DISP1   Abs  997199 #F374F -   236   230
 DISP2   Abs  997619 #F38F3 -   455   276
 DISP25  Abs  997645 #F390D -   477   467
 DISP30  Abs  997737 #F3969 -   510   494
 DISPEX  Abs  997755 #F397B -   527   201   314   326   416   452   507
 DISPEx  Abs  997369 #F37F9 -   314   312   336
 DISPN.  Abs  997080 #F36D8 -   171   130   137
 DISPN1  Abs  997120 #F3700 -   189   149   171
 DISPN0  Abs  997003 #F368B -   118    99   101
 DISPNS  Abs  997000 #F3688 -   114    90   102   110
 DISP0F  Abs  997840 #F39D0 -   568    79
 DISP0K  Abs  997165 #F372D -   210   111   183
 DISP0X  Abs  997834 #F39CA -   559   204   351   386   474   527   541   554
                              556
 DISP0x  Abs  997161 #F3729 -   204   225   267   280
 DISPn3  Abs  997060 #F36C4 -   161   153
 DISPn4  Abs  997074 #F36D2 -   166   132   158   164
 DISPnE  Abs  997255 #F3787 -   266   217
 DISPoF  Abs  996937 #F3649 -    79    87
 DISPoX  Abs  997641 #F3909 -   474   461   499
 DISPox  Abs  997431 #F3837 -   351   347
 DSPBFS  Ext                -   287   300   766
 DSPCL?  Ext                -   480
 DSPSET  Ext                -    82   173   483   537
 DSPSTA  Ext                -   477   481   569   953
 DelChr  Abs  997373 #F37FD -   320   244   246
 DelEx   Abs  997609 #F38E9 -   450   440
 DelL0   Abs  997531 #F389B -   407   402
 DelL1   Abs  997556 #F38B4 -   422   406
 DelLin  Abs  997510 #F3886 -   397   250
 Delete  Abs       4 #00004 -    67   236   324   519   634   677   713   746
 Delexc  Abs  997552 #F38B0 -   416   409
 DispOK  Ext                -    91   114   131   172   536
 DspErr  Abs  997775 #F398F -   535   528   552
 DspSn0  Abs  997802 #F39AA -   549   232
 DspSn1  Abs  997806 #F39AE -   550   534
 DspSn2  Abs  997820 #F39BC -   555   522
```

```
 DspSnd  Abs  997799 #F39A7 -   544   515
 DspsnO  Abs  997192 #F3748 -   231   270   275   282   337
 END     Ext                -   564
 ESCSTA  Ext                -   214
 Esc     Abs      27 #0001B -    62   459   965
 EscSnd  Abs  997761 #F3981 -   531   256
 FINDA   Ext                -   238
 FNDMBX  Ext                -   109
 FarEnd  Abs  997491 #F3873 -   383   381
 FarLft  Abs  997504 #F3880 -   389   254
 FarRt   Abs  997441 #F3841 -   362   252
 FarRt1  Abs  997453 #F384D -   370   378
 FarRt2  Abs  997470 #F385E -   376   371
 FarRt3  Abs  997480 #F3868 -   379   375
 Farxx   Abs  997444 #F3844 -   367   391
 GETMBX  Ext                -   307   333   407   503   531   549   661
 GETMSK  Ext                -   837
 GTYPE   Ext                -   136
 IS-DSP  Ext                -    75   177   191
 InsChr  Abs  997384 #F3808 -   329   248
 Insert  Ext                -   498   514   632   690   759
 LArrow  Abs  997435 #F383B -   354   242   471
 LCleft  Abs  998379 #F3BEB -  1017   446   778   860
 LoopOK  Ext                -    86   535   878
 MOVC10  Abs  998150 #F3B06 -   830   843
 MOVC15  Abs  998212 #F3B44 -   854   852
 MOVC17  Abs  998228 #F3B54 -   860   901
 MOVC20  Abs  998235 #F3B5B -   862   856
 MOVC30  Abs  998252 #F3B6C -   875   865
 MOVC40  Abs  998266 #F3B7A -   879   877
 MOVC45  Abs  998268 #F3B7C -   880   871
 MOVC50  Abs  998272 #F3B80 -   884   834
 MOVC55  Abs  998292 #F3B94 -   904   886
 MOVC60  Abs  998301 #F3B9D -   911   831
 MOVC70  Abs  998310 #F3BA6 -   917   912
 MOVCU+  Abs  998139 #F3AFB -   826   377
 MOVCUR  Abs  998136 #F3AF8 -   825   350   558
 MTYL    Ext                -   170
 NotEnd  Abs  997875 #F39F3 -   636   628   633
 NotPro  Abs  997900 #F3A0C -   651   649
 PUTD    Ext                -   966
=PUTEsc  Abs  998336 #F3BC0 -   965   408   532   679   696   761
 PUTX    Ext                -   335   506
 Printr  Ext                -   100   145   157   224   272
 Protec  Abs       6 #00006 -    70   645   650   670   721   755
 Putd    Abs  998340 #F3BC4 -   966   313   411   551   682   699   729   749
                               763
 RArrow  Abs  997409 #F3821 -   340   240
 RepCur  Abs       0 #00000 -    65   376   825   851
 SCNRT   Ext                -   397   622
 SENDI+  Ext                -   439   876
 SENDIT  Ext                -   449   780
 SETLP   Ext                -   108
 START   Ext                -   129
 Send#   Abs  997974 #F3A56 -   707   669   691
```

Send+	Abs	997941	#F3A35	-	681	678					
Send-	Abs	997951	#F3A3F	-	690	671					
SendO0	Abs	997990	#F3A66	-	713	711					
SendO2	Abs	998010	#F3A7A	-	728	720	722				
=SendBf	Abs	997855	#F39DF	-	618	325	520				
SendL1	Abs	998071	#F3AB7	-	764	756	758	760			
SendLs	Abs	998044	#F3A9C	-	755	747					
SendNI	Abs	998019	#F3A83	-	734	714					
Sendex	Abs	997947	#F3A3B	-	683	635	680	697	700	730	742	750
SetCur	Abs	6	#00006	-	69	70	231	349	369	521	533	548
					553	864	885					
WRITIT	Ext			-	308	738					
Wallby	Ext			-	98	144	165	668	757			

Input Parameters

  Source file name is NZ&DSP::MS

  Listing file name is NZ/DSP:TI:ML::-1

  Object file name is NZ%DSP:TI:MS::-1

```
                                  111111
                        0123456789012345
```
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1                *
 2                *       N   N  ZZZZZ   &     BBBB   U   U  TTTTT
 3                *       N   N      Z  & &    B   B  U   U    T
 4                *       NN  N     Z   & &    B   B  U   U    T
 5                *       N N N    Z     &     BBBB   U   U    T
 6                *       N  NN   Z     & & &  B   B  U   U    T
 7                *       N   N  Z      &  &   B   B  U   U    T
 8                *       N   N  ZZZZZ   && &  BBBB    UUU     T
 9                *
10                *
11                        TITLE  BASIC UTILITIES <840104.1515>
12 F3BF7                  ABS    #F3BF7        TI%HP6 address (fixed)
13                ********************************************************************
14                ********************************************************************
15                **
16                ** Name:       GETMBX - Get address of mailbox (last FNDMBX)
17                **
18                ** Category:   PTRUTL
19                **
20                ** Purpose:
21                **      Get the HPIL mailbox address from RAM and put it in D0
22                **
23                ** Entry:
24                **      Nothing
25                **
26                ** Exit:
27                **      C[A], D0-->Mailbox
28                **      Carry clear
29                **
30                ** Calls:      None
31                **
32                ** Uses.......
33                **  Inclusive: C[A],D0
34                **
35                ** Stk lvls:   0
36                **
37                ** NOTE: Does not alter P!
38                **
39                ** History:
40                **
41                **    Date     Programmer            Modification
42                **  --------   ----------   ---------------------------------
43                **  11/11/82      NZ        Added documentation
44                **
45                ********************************************************************
46                ********************************************************************
47 F3BF7 1B00     =GETMBX D0=(5) =MBOX^        Mailbox pointer (in RAM)
         000
48 F3BFE 146              C=DAT0 A             Read the pointer to the mailbox
49                *
50 F3C01 F2               CSL    A             Mbox address is stored as words
51                *                              offset from 20000!
52 F3C03 80F4             CPEX   4
53 F3C07 22               P=     2
54 F3C09 80F4             CPEX   4             Set nibble 4 to 2 (page 20000)
```

```
                *
                * Now C[A] is the mailbox address
                *
F3C0D 134               D0=C                    Put the address into D0...
F3C10 03                RTNCC                   ...and return with carry clear!
                ********************************************************************
                ********************************************************************
                **
                ** Name:        SETLP - Set up C[S] for FNDMBX from D[A] info
                **
                ** Category:    PILUTL
                **
                ** Purpose:
                **      Given D[A] set up for device search, return the loop #
                **      minus one in C[S]
                **
                ** Entry:
                **      D[A] is device info (see START documentation)
                **
                ** Exit:
                **      Carry clear
                **      P=0
                **      Mailbox # in C[S]
                **
                ** Calls:       None
                **
                ** Uses.......
                **  Inclusive: C[A],C[S],P
                **
                ** Stk lvls:    0
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------    --------------------------------
                **  11/11/82       NZ         Added documentation
                **
                ********************************************************************
                ********************************************************************
F3C12 20    =SETLP  P=      0
F3C14 310E          LCHEX   E0
F3C18 0E6F          C=C!D   B               To check for FIND (Could be [A])
F3C1C B66           C=C+1   B               If carry, is a FIND...
                *
                * If carry, FIND some device...if not carry, address
                * (If address, high bits of D[XS]=mailbox #; else D[3]=mbox #)
                *
F3C1F DB            C=D     A               Copy to C[A] for either case
F3C21 570           GONC    SETLP1          Go if address
F3C24 F6            CSR     A
F3C26 490           GOC     SETLP2          Go always (FIND Nth device)
                *-
                *-
                *
                * Address!
```

```
                *
F3C29 BB6  SETLP1  CSR    X            (Clears C[XS])
F3C2C C6           C=C+C  A            Multiply C[X]*4
F3C2E C6           C=C+C  A            Now C[2] is the mailbox #
F3C30 80D2 SETLP2  P=C    2            Now P is the mailbox #...
F3C34 80CF         C=P    15           ...now in C[S]!
F3C38 20           P=     0
F3C3A 03           RTNCC
               ****************************************************************
               ****************************************************************
               **
               ** Name:       FNDMBX - Find an HPIL mailbox (C[S] is #)
               ** Name:       FNDMB- - Find mailbox, clear disp bits, chk OFF
               ** Name:       FNDMBD - Find an HPIL mailbox, clear disp bits
               ** Name:       FNDMB+ - Find an HPIL mailbox (D[A] is spec)
               **
               ** Category:   PTRUTL
               **
               ** Purpose:
               **      Search the configuration tables to find a HPIL mailbox
               **      (C[S] is the number of the mailbox minus 1 - if C[S]
               **      is 2 then find the 3rd mailbox!)
               **
               ** Entry:
               **      FNDMBX,FNDMB-,FNDMBD:
               **        C[S] is the mailbox number -1
               **      FNDMB+:
               **        D[A] is the device spec
               **
               ** Exit:
               **      Carry clear: D0 points to the mailbox, (MBOX^) is set
               **            to the mailbox
               **      Carry set: Mailbox and/or configuration buffer not
               **            found (P is the error number)
               **
               ** Calls:      CNFFND  (FNDMB+ also calls SETLP)
               **
               ** Uses.......
               **  Exclusive: C[W],D0,P
               **  Inclusive: C[W],D0,P
               **
               ** Stk lvls:   1 (CNFFND)(SETLP)
               **
               ** History:
               **
               **    Date     Programmer          Modification
               **  --------   ----------   ---------------------------------
               **  05/23/83      NZ        Reworked error exit for loop is
               **                          now "OFFED" (Returns P= =eOFFED)
               **  03/16/83      NZ        Changed FNDMBe to return P=eBADMD
               **  03/08/83      NZ        Added FNDMB-
               **  11/11/82      NZ        Added documentation
               **
               ****************************************************************
               ****************************************************************
```

```
165                *
166                * First set C[S] to be the mailbox #, minus 1
167                *
168 F3C3C 72DF =FNDMB+ GOSUB  SETLP
169                *
170                * C[S] is now the mailbox #
171                *
172 F3C40      =FNDMB-
173                *
174                * Get LOOP STatus to clear InptOK bit
175                *
176 F3C40 1B00         DO=(5) =LOOPST
           000
177 F3C47 1562         C=DATO XS           Read into ST[3:0]
178 F3C4B 0B           CSTEX
179                *
180                * Is the following test desirable??? (will error out if OFFED)
181                *
182 F3C4D 20           P=     =eOFFED      Set P before the test
183 F3C4F 870          ?ST=1  =Offed       Is the loop "OFFED" (OFFIO)?
184 F3C52 20           GOYES  FNDMB.       Set carry if "RESTORE IO Needed"
185 F3C54      FNDMB.
186 F3C54 0B           CSTEX
187 F3C56 4A6          GOC    FNDMB9       "RESTORE IO Needed"
188 F3C59 D2           C=0    A            Clear "set up" bits, "Device" bit
189 F3C5B 1542         DATO=C XS           Device, "set up" bits cleared
190                *
191                * Set DispOK bit false (Display is NOT set up on loop)
192                *
193 F3C5F 1B00 =FNDMBD DO=(5) =DSPSET
           000
194 F3C66 1562         C=DATO XS
195 F3C6A 0B           CSTEX
196 F3C6C 840          ST=0\  =DispOK      Display is NOT set up
197 F3C6F 0B           CSTEX
198 F3C71 1542         DATO=C XS
199                *
200                * Get the mailbox address (search the device table for it)
201                *
202 F3C75 80DF =FNDMBX P=C    15           Save mailbox # in P for now
203 F3C79 D6           C=A    A
204 F3C7B 7974         GOSUB  Cslc5        Save A[A] in C[9:5]
205 F3C7F 80CF         C=P    15           Restore mailbox # to C[S]
206 F3C83 137          CD1EX               SAVE D1 IN DO (TEMPORARILY)
207 F3C86 134          DO=C
208 F3C89 20           P=     0
209 F3C8B 31DF         LCHEX  FD           CONFIGURATION BUFFER - MM I/O
210 F3C8F 8F00         GOSBVL =CNFFND      Configuration find
           000
211 F3C96 542          GONC   FNDMBE       ...Not found (error!!!)
212                *
213                * Found memory-mapped i/o buffer!!!!
214                *
215 F3C99 173          D1=D1+ 4            Skip to proper offset into entry
216 F3C9C 24           P=     4
```

```
F3C9E 8A8  FNDMB1  ?A=0   A            Done searching yet?
F3CA1 A1           GOYES  FNDMBE       Yes...didn't find a mailbox!
F3CA3 147          C=DAT1 A
F3CA6 A0E          C=C-1  P            If zero, is PIL mailbox!
F3CA9 452          GOC    FNDMB3       Yep...found it!
                   *
                   * Haven't found it yet...keep trying!
                   *
F3CAC 179  FNDMB2  D1=D1+ 10           Next entry
F3CAF 132          ADOEX
F3CB2 189          DO=DO- 10           Decrement A[A] by 10
F3CB5 132          ADOEX
F3CB8 55E          GONC   FNDMB1       Loop back for more...
                   *
                   * This is an error!
                   *
F3CBB 20   FNDMBE  P=     0
F3CBD 0D           P=P-1               Set carry!!!
F3CBF 20           P=     =eNMBOX      No mailbox...carry is set!
                   *
                   * Restore A[A], D1 before returning (COMMON return code)
                   *
F3CC1 136  FNDMB9  CDOEX               Old D1 value-->C, C[A] to DO
F3CC4 135          D1=C                Now D1 is restored
F3CC7 7424         GOSUB  Csrc5
F3CCB DA           A=C    A            Now A[A] is restored
F3CCD 01           RTN
                   *-
                   *-
F3CCF      FNDMB3
                   *
                   * Found a mailbox...check if it is the correct one!
                   *
F3CCF A4E          C=C-1  S
F3CD2 59D          GONC   FNDMB2       Go to the next entry!
                   *
                   * Have THE mailbox!
                   * (P is still 4)
                   *
                   * Save the address away in MBOX^ first!
                   *
F3CD5 1F00         D1=(5) =MBOX^
      000
F3CDC 15D2         DAT1=C 3
                   *
                   * Now get the actual address
                   *
F3CE0 F2           CSL    A            Offset to words (multiply by 16)
F3CE2 302          LCHEX  2            Now C has the mailbox address!
F3CE5 21           P=     1
F3CE7 0D           P=P-1               Clear carry, set P=0
F3CE9 57D          GONC   FNDMB9       GO ALWAYS!
                   ********************************************************************
                   ********************************************************************
                   **
```

```
                ** Name:       CHKASN - Check if an HPIL assignment is active
                **
                ** Category:   PILUTL
                **
                ** Purpose:
                **      Check if the assignment is none, HPIL, or "other"
                **      (If "OFF"ed, returns as if no assignment)
                **
                ** Entry:
                **      C[6:0] is the assignment table value
                **
                ** Exit:
                **      Carry set if not assigned/not HPIL/"OFF"ed/LOOP/NULL
                **      Carry clear if assigned...B[W],C[X] set up for START
                **        If C[S]<>0, this is a FIND (Address unknown)
                **
                ** Calls:      I/OFND
                **
                ** Uses.......
                **  Exclusive: B[W],C[W],P
                **  Inclusive: B[W],C[W],P
                **
                ** Stk lvls:   2 (pushed D1;I/OFND)
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **  --------   ----------    ---------------------------------
                **  11/11/82      NZ         Added documentation
                **
                ********************************************************************
                ********************************************************************
F3CEC           =CHKASN
                *
                * Assign table format:
                *
                *       nib:   usage:
                *       ---    ------
                *       2-0:   If device address known, address, loop # here
                *              If LOOP, nibs 1-0=0, nib 2 is loop #
                *              If NULL, F00
                *              If not known/not assigned/iobuffer, FFF
                *              If assigned, not HPIL, Fxx, xx<>FF
                *
                *         3:   If unassigned/not HPIL, F
                *              If IO buffer with one entry, 4
                *              If address specified, 0
                *              If type specified, loop # + 1 (nib 3: 1,2,3)
                *              If this assignment has been "OFF"ed, bit 3 is 1
                *
                *       6-4:   If type, nib 6: sequence #, nibs 5-4: Acc id
                *              If address, 6-4: address, loop #
                *              If IO buffer, 6-4: io buffer #
                *              If unassigned (NOT "OFF"ed), FFF
                *              If not HPIL and nib 3=F, not defined
```

```
                *
F3CEC AC2       C=0   S          Preclear "FIND" flag
F3CEF 20        P=    0
                *
                * Check if this is OK as is...
                *
F3CF1 96A       ?C=0  B          Is it LOOP or NULL?
F3CF4 00        RTNYES           Yes..."not" set up
F3CF6 B26       C=C+1 XS
F3CF9 A2E       C=C-1 XS
F3CFC 5D0       GONC  CHKAS#     This is OK as is unless OFFED
F3CFF B36       C=C+1 X
F3D02 A3E       C=C-1 X
F3D05 4F0       GOC   CHKAS0
F3D08 02        RTNSC            This is NOT a HPIL assignment!
                *-
                *-
F3D0A 0B  CHKAS# CSTEX
F3D0C 87B       ?ST=1 11         Is this offed (carry if so)
F3D0F 20        GOYES CHKAS!
F3D11 0B  CHKAS! CSTEX
F3D13 01        RTN              Carry indicates state!
                *-
                *-
F3D15     CHKAS0
                *
                * Check if this is not assigned (nibble 3="F")
                *
F3D15 23        P=    3
F3D17 B06       C=C+1 P
F3D1A A0E       C=C-1 P          Alter carry only...not value!
F3D1D 20        P=    0          Reset P to 0!
F3D1F 400       RTNC             Not defined...return!
F3D22 BF6       CSR   W          Now code nibble in C[XS]
F3D25 92E       ?C#0  XS
F3D28 60        GOYES CHKAS1     This is not an address...
F3D2A 6470      GOTO  CHKAS9     This is an address!
                *-
                *-
                *
                * If here, have either iobuffer, type, or "OFF"ed assignment
                *
F3D2E BF6 CHKAS1 CSR  W          C[1] is the code nibble!
F3D31 80D1      P=C   1          Copy C[1] into P
F3D35 80CF      C=P   15         Use C[S] to test it
                *
                * If C[S] is >=8, then "OFF"ed (RTNSC)
                *
F3D39 A46       C=C+C S
F3D3C AC2       C=0   S          Clear it again!
F3D3F 20        P=    0
F3D41 400       RTNC             If carry, "OFF"ed!
                *
                * Now either iobuffer or type
                *
```

```
    381 F3D44 80D1          P=C    1
    382 F3D48 890           ?P=    =SngDev        Is this a single entry buffer?
    383 F3D4B 71            GOYES  CHKAS2         Yes...process it!
    384               *
    385               * This is a TYPE!
    386               *
    387 F3D4D F6            CSR    A
    388 F3D4F F6            CSR    A              C[XS] is sequence #, C[B] is type
    389 F3D51 D5            B=C    A              Copy to B[B]
    390               *
    391               * C[XS] is sequence #, P is loop # + 1 (C[4:3]=0)
    392               *
    393 F3D53 0D            P=P-1                 P is now loop #
    394 F3D55 80F3          CPEX   3              Get loop # in C[3]
    395 F3D59 A4E           C=C-1  S              Set C[S]="F" for "FIND" flag
    396               *
    397               * Now C[3] is loop #, C[XS] is sequence #, P=0
    398               *
    399 F3D5C 3100          LC(2)  =DevTyp        This is a device type!
    400 F3D60 03            RTNCC                 C[2] is seq #, B[B] is ACC ID
    401               *                           C[3] is loop #
    402               *-
    403               *-
    404 F3D62       CHKAS2
    405               *
    406               * I/O buffer!
    407               *
    408               * C[4:2] is I/O buffer #
    409               *
    410               * Now save A[W] in B[W], D1 on RSTK
    411               *
    412 F3D62 7540          GOSUB  CHKASs         Save info, find the buffer
    413 F3D66 563           GONC   CHKASx         Not found...(Error!)
    414               *
    415               * Now D1 @ I/O buffer start, A[A] is length of buffer
    416               *
    417 F3D69 147           C=DAT1 A              Read type, seq #, loop #
    418 F3D6C 172           D1=D1+ 3              Move to next word
    419 F3D6F 1537          A=DAT1 W
    420 F3D73 AFC           ABEX   W              Restore A[W], B[W] is ID/label
    421 F3D76 816           CSRC                  Type in C[S] now
    422 F3D79 F2            CSL    A
    423 F3D7B F2            CSL    A
    424               *
    425               * Now C[3] is loop #, C[2] is sequence #
    426               *
    427               *
    428               * P is now zero...clear C[S], set P=C[S]
    429               *
    430 F3D7D 80FF          CPEX   15             Find out what it is
    431               *
    432               * P is now device type
    433               *
    434 F3D81 137           CD1EX
    435 F3D84 1D00          D1=(2) =DevID         Preload Device ID
```

```
436 F3D88 892          ?P=    2             Device ID?
437 F3D8B 60           GOYES  CHKAS3        (Set carry if Device ID)
438 F3D8D 1D00         D1=(2) =VolLbl       Volume label!
439 F3D91       CHKAS3
440 F3D91 A4E          C=C-1  S             Set C[S]="F"
441 F3D94 20   CHKAS4  P=     0
442 F3D96 07           C=RSTK               Restore D1
443 F3D98 137          CD1EX
444 F3D9B 03           RTNCC                Done (return, carry clear)
445             *-
446             *-
447 F3D9D 02   CHKASx  RTNSC
448             *-
449             *-
450             *
451             * This is an address!
452             *
453 F3D9F       CHKAS9
454 F3D9F BF6          CSR    W             (Clears C[S])
455 F3DA2 A4E          C=C-1  S             Set C[S]="F" (Do store on return)
456 F3DA5 F6           CSR    A
457 F3DA7 F6           CSR    A             Now C[X] is the address!
458 F3DA9 03           RTNCC
459             *-
460             *-
461 F3DAB F6   CHKASs  CSR    A
462 F3DAD F6           CSR    A             Shift the ID to C[X]
463 F3DAF 20           P=     0             Set P=0 for later!
464 F3DB1 D5           B=C    A
465 F3DB3 07           C=RSTK               Save calling address in B[A]
466 F3DB5 137          CD1EX
467 F3DB8 06           RSTK=C               Save D1 on RSTK
468 F3DBA 137          CD1EX
469 F3DBD 06           RSTK=C               Restore calling routine address
470 F3DBF D9           C=B    A             Restore C[A]
471 F3DC1 AF8          B=A    W             Save A in B[W]
472 F3DC4 6943         GOTO   1/OFND        Find it!
473             ********************************************************************
474             ********************************************************************
475             **
476             ** Name:       SETUP - Given info from START, set up C[6:0]
477             **
478             ** Category:   PILUTL
479             **
480             ** Purpose:
481             **      Build a recall string in C[6:0] (carry set if buffer
482             **      required to store this)
483             **
484             ** Entry:
485             **      D is the info returned from START
486             **        D[X] is address, (loop #) * 1024
487             **        D[S] is type (0=address, 1=device type, 2=device ID,
488             **        3=volume label, 4=NULL, 5=LOOP)
489             **        D[3] is sequence # for types 1 and 2
490             **      B is as returned from START
```

```
491             **
492             ** Exit:
493             **      C[6:0] is the information to put into an IS-xxx entry
494             **      P=0
495             **      C[S]=0 if entry will fit in IS-xxx, else C[S]#0
496             **
497             ** Calls:      CSLC5,CSRC4,CSLC3
498             **
499             ** Uses.......
500             **  Inclusive: C[W],P
501             **
502             ** Stk lvls:   1 (CSLC5)(CSRC4)(CSLC3)
503             **
504             ** History:
505             **
506             **    Date     Programmer             Modification
507             **  --------   ----------   ------------------------------
508             **  04/22/83      NZ        Fixed bug of creating an I/O buf
509             **                          for NULL and LOOP
510             **  11/12/82      NZ        Added documentation
511             **
512             ****************************************************************
513             ****************************************************************
514 F3DC8       =SETUP
515             ****************************************************************
516             *
517             * D[S] is type:
518             *       0: Address
519             *       1: Device type, sequence #
520             *       2: Device ID, sequence #
521             *       3: Volume label
522             *       4: NULL
523             *       5: LOOP
524             *
525             * Buffer layout:
526             *   +-----------------------------------------------------+
527             *   | Device ID/vol Lbl | search type | loop # | sequence # |
528             *   +-----------------------------------------------------+
529             * nibs:      16               1          1         1
530             * (high memory)                                (low memory)
531             *
532             *
533             * Layout of entry:
534             *  Type=0,4,5: (For types 4 & 5, true addr = 0)
535             *   +-----------------------------------------------------+
536             *   | Find address + loop*1024 | 0 | true addr + loop*1024 |
537             *   +-----------------------------------------------------+
538             * nibs:            3             1             3
539             * (high memory)                                (low memory)
540             *
541             *  Type=1:
542             *   +-----------------------------------------------------+
543             *   | Seq # | device type | loop + 1 | true addr+loop*1024 |
544             *   +-----------------------------------------------------+
545             * nibs: 1          2          1               3
```

```
546                *
547                *
548                ****************************************************************
549                *
550 F3DC8 DB               C=D    A            Copy address first to save code
551 F3DCA ACB              C=D    S            Get device type from D[S]
552 F3DCD A4E              C=C-1  S
553 F3DD0 441              GOC    SETUP0       Address
554 F3DD3 A4E              C=C-1  S            Is it a device type (acc id)?
555 F3DD6 4E2              GOC    SETUP1       Yes...continue
556 F3DD9 A4E              C=C-1  S            Is it a device ID?
557 F3DDC 464              GOC    SETUP2       Yes...continue
558 F3DDF A4E              C=C-1  S            Is it a volume label?
559 F3DE2 404              GOC    SETUP2       Yes...continue
560                *
561                * This is either address, NULL, or LOOP
562                *
563 F3DE5 7F03 SETUP0  GOSUB  Cslc5         Rotate 5 nibbles (so C=0 A works)
564 F3DE9 D2               C=0    A            Clear device type (address=0)
565 F3DEB BF6              CSR    W
566 F3DEE ABB  SETUPx  C=D    X            Now C[6:0] is set up!
567 F3DF1 2F               P=     15
568 F3DF3 300              LC(1)  =DsNull      Check if NULL
569 F3DF6 20               P=     0
570 F3DF8 947              ?C#D   S
571 F3DFB 50               GOYES  SETUP,       Not NULL
572 F3DFD A2E              C=C-1  XS           NULL...set C[X]="FOO"
573 F3E00 AC2  SETUP,  C=0    S            Clear flag for WILL fit...
574 F3E03 03               RTNCC               Return...WILL fit in entry!
575                *-
576                *-
577 F3E05      SETUP1
578                *
579                * Device type
580                *
581 F3E05 AD2              C=0    M            Clear high nibbles of C[A]
582 F3E08 C6               C=C+C  A
583 F3E0A C6               C=C+C  A            C[3] is now loop #
584 F3E0C 72E2             GOSUB  Csrc4        Put loop # into C[S]
585 F3E10 DB               C=D    A            C[3] is sequence # now
586 F3E12 F6               CSR    A            C[2] is sequence #
587 F3E14 AE9              C=B    B            C[X] is sequence #, type
588 F3E17 B46              C=C+1  S            Now loop + 1 in C[S]
589 F3E1A 8E00             GOSUBL =CSLC4       C[6:4] is seq #, type; C[3]=loop
            00
590 F3E20 5DC              GONC   SETUPx       Go always!
591                *-
592                *-
593 F3E23      SETUP2
594                *
595                * Whether this is a device ID or a volume label, the following
596                * is all the same!
597                *
598 F3E23 D2               C=0    A
599 F3E25 AAB              C=D    XS           Loop*4 to C[XS]
```

```
600 F3E28 C6          C=C+C  A
601 F3E2A C6          C=C+C  A            Now loop # in C[3]
602 F3E2C F2          CSL    A            Loop # to C[4]
603 F3E2E 23          P=     3
604 F3E30 A8B         C=D    P            Copy D[3] (sequence #)
605 F3E33 BF2         CSL    W            Loop # to C[5], seq # to C[4]
606 F3E36 ABB         C=D    X            Copy true address to C[X]
607 F3E39 2F          P=     15
608 F3E3B 304         LC(1)  4            Offset from D[S] value for table
609 F3E3E A4B         C=C+D  S
610 F3E41 80FF        CPEX   15           Set C[S]="F", P=type+4
611 F3E45 80F3        CPEX   3            Store type in C[3], set P=0
612 F3E49 03          RTNCC               Return, C[S]="F" (won't fit)
613           ****************************************************************
614           ****************************************************************
615           **
616           ** Name:       SAVEIT - Save device info at (D1) (7 nibbles)
617           **
618           ** Category:   PILUTL
619           **
620           ** Purpose:
621           **      Save device descripter entry @ D1
622           **
623           ** Entry:
624           **      D1 @ destination entry
625           **      B,C are exit conditions of SETUP
626           **
627           ** Exit:
628           **      Carry clear, P=0 (Error exits directly)
629           **
630           ** Calls:      CSRC3;4;5,CSLC4;9,I/OALL,I/OFSC,I/ODAL
631           **
632           ** Uses.......
633           **  Exclusive: A,B,C,D,R2,R3,D0,D1,P
634           **  Inclusive: A,B,C,D,R2,R3,D0,D1,P
635           **
636           ** Stk lvls:   3 (I/OALL)(I/ODAL)
637           **
638           ** Algorithm:
639           **              Check if entry will fit in 7 nibbles:
640           **              If will not fit, goto SAVEI1
641           **      SAVEI0:Read old entry; write new entry
642           **              If old entry used buffer, deallocate the buffer
643           **              RTNCC
644           **             ----
645           **      SAVEI1:Create a buffer for the entry
646           **              Write the entry
647           **              Build the info for the 7 nibble field
648           **              Goto SAVEI0
649           **
650           ** History:
651           **
652           **    Date     Programmer             Modification
653           **  --------   ----------    ---------------------------------
654           **  07/21/83      NZ         Changed error exit to direct exit
```

```
                        **  11/12/82      NZ          Added documentation
                        **
                        ****************************************************************
                        ****************************************************************
F3E4B 94E   =SAVEIT ?C#0    S               Does this need an I/O buffer?
F3E4E 92            GOYES   SAVEI1          Yes...needs I/O buffer!
                        *
                        * Will fit in IS-xxx entry...write it!
                        *
F3E50 15B6  SAVEI0  A=DAT1  7               Read old type...
F3E54 15D6          DAT1=C  7               ...write new type...
                        *
                        * Now check if old type used an I/O buffer
                        *
F3E58 AF6           C=A     W               Must be WORD for CSRC4 below!!!
F3E5B 80D3          P=C     3               Check code nibble
F3E5F 890           ?P=     =SngDev
F3E62 20            GOYES   SAVEI-          Single item I/O buffer
F3E64 20    SAVEI-  P=      0
F3E66 500           RTNNC                   Done if no carry!
F3E69 7582          GOSUB   Csrc4           Buffer # in C[X] now
F3E6D 8E00          GOSUBL  =I/odal         Deallocate the buffer
          00
F3E73 20            P=      0
F3E75 03            RTNCC
                        *-
                        *-
F3E77       SAVEI1
                        *
                        * Will NOT fit in IS-xxx entry...create a buffer &write it out
                        *
                        * C[X] is true address, C[4] is sequence #, C[5] is loop #
                        * D[S] is type
                        *
F3E77 7772          GOSUB   Csrc4
F3E7B ACB           C=D     S               Save D[S] in C (--> R2)
F3E7E 8E00          GOSUBL  =CSLC9          C[8:5] is type, addr
          00
F3E84 137           CD1EX
F3E87 10B           R3=C                    Save D1 in R3[A],info in R3[11:5]
F3E8A AF9           C=B     W
F3E8D 10A           R2=C                    Save B[W] in R2
F3E90 8E00          GOSUBL  =I/OFSC         Find I/O scratch buffer
          00
F3E96 425           GOC     NORAMe          Error...no buffers (eMEM?)
                        *
                        * Now Buffer ID in C[X]
                        *
F3E99 D5            B=C     A               Save ID in B[X]
F3E9B D2            C=0     A
F3E9D 3131          LC(2)   19              Need 19 (decimal) nibs for it!
F3EA1 DD            BCEX    A
F3EA3 8F00          GOSBVL  =I/OALL         Allocate a buffer for this!
          000
F3EAA 5E3           GONC    NORAMe          Error (eMEM?)
```

```
706                *
707                * D1 @ data start, D0 @ buffer ID
708                *
709 F3EAD 11B              C=R3                Recover the first info
710 F3EB0 135              D1=C                Recover pointer to save area
711 F3EB3 7832             GOSUB  Csrc5        Move address to C[X]
712 F3EB7 D7               D=C    A            Save address in D[X]
713 F3EB9 8E00             GOSUBL =CSRC3       Move Loop, seq #, type to C[X]
            00
714 F3EBF 1523             A=DAT0 X            Read buffer ID...
715 F3EC3 165              D0=D0+ 6            ...point to data area...
716 F3EC6 15C2             DAT0=C 3            Write out the seq #, loop, type
717 F3ECA 162              D0=D0+ 3            Point to next area
718 F3ECD 11A              C=R2                Recall dev word
719 F3ED0 1547             DAT0=C W            Write out the device ID/vol label
720                *
721                * Now set up the descriptor entry!
722                *
723 F3ED4 AB6              C=A    X            Get buffer ID
724 F3ED7 7022             GOSUB  Cslc4        ID --> C[6:4]
725 F3EDB 23               P=     3
726 F3EDD 300              LC(1)  =SngDev      Single item buffer
727 F3EE0 20               P=     0
728 F3EE2 ABB              C=D    X            Recall address!
729 F3EE5 6A6F             GOTO   SAVEIO       Write it out, check old buffer
730                *-
731                *-
732 F3EE9 20    =NORAMe    P=     =eNORAM      Insufficient memory
733 F3EEB 8C00             GOLONG =ERRORX      Set it up!
            00
734                ****************************************************************
735                ****************************************************************
736                **
737                ** Name:       RESTOR - Clear "OFFED" bits in IS table entries
738                **
739                ** Category:   PILUTL
740                **
741                ** Purpose:
742                **      Reactivate all devices (clear their OFFED bits)
743                **
744                ** Entry:
745                **      Nothing
746                **
747                ** Exit:
748                **      Carry clear
749                **
750                ** Calls:      Nothing
751                **
752                ** Uses.......
753                **  Inclusive: C[XS],D0
754                **
755                ** Stk lvls:   1 (Internal GOSUB)
756                **
757                ** NOTE: Does not alter P!
758                **
```

```
                ** History:
                **
                **    Date     Programmer          Modification
                **  --------   ----------   --------------------------------
                **  11/12/82      NZ        Added documentation
                **
                ****************************************************************
                ****************************************************************
F3EF1 1B00 =RESTOR D0=(5) (=IS-DSP)+3   IS-DSP+3
      000
F3EF8 7300        GOSUB  PILCNs
F3EFC 166         D0=D0+ 7              IS-PRT+3
                *
                * Fall into PILCHs for IS-PRT, return carry clear
                *
F3EFF 1562 PILCNs  C=DAT0 XS
F3F03 B26         C=C+1  XS
F3F06 A2E         C=C-1  XS
F3F09 4D0         GOC    PILCns         If "Fxx", leave as is!
F3F0C 0B          CSTEX
F3F0E 84B         ST=0   11             Clear OFFed flag
F3F11 0B          CSTEX
F3F13 1542        DAT0=C XS
F3F17 03   PILCns  RTNCC
                ****************************************************************
                ****************************************************************
                **
                ** Name:       GETSTR - Set up for string/literal expression
                **
                ** Category:   EXCUTL
                **
                ** Purpose:
                **      Set up either a literal or string expression
                **
                ** Entry:
                **      D0 points to the item in memory
                **
                ** Exit:
                **      If error, takes hard error exit (EXPEXC, REVPOP)
                **      Carry clear
                **      ST(=sSTK)=0: D0 points to the first character
                **      ST(=sSTK)=1: D[A] is the end of the string
                **                   D1 points to the first character
                **                   A[A] is the string length in nibbles
                **
                ** Calls:      EXPEX+,RESTST,REVPOP,D1=AVE
                **
                ** Uses.......
                **  Exclusive: A,  C,D,               D0,D1,P,       ST[sSTK]
                **  Inclusive: A,B,C,D,R0,R1,R2,R3,R4,D0,D1,P,FUNCxx,ST[11:0]
                **
                ** Stk lvls:   5 (EXPEX+)
                **
                ** History:
                **
```

```
                **    Date     Programmer            Modification
                **  --------   ----------   ------------------------------
                **  03/16/83      NZ        Changed EXPEXC to EXPEX+, added
                **                          call to RESTST
                **  11/12/82      NZ        Added documentation
                **
                ****************************************************************
                ****************************************************************
F3F19 840   =GETSTR ST=0   =sSTK
F3F1C 14A           A=DAT0 B              Read in the first character
                *
                * Check first if this is t*!
                *
F3F1F 20            P=     0
F3F21 3100          LC(2)  =tCOLON        Check if device spec, no filename
F3F25 962           ?A=C   B              Is this device spec?
F3F28 83            GOYES  GETST1         Yes...exit, sSTK=0, D0 @ tCOLON
                *
                * This is not a literal device spec...check literal file spec
                *
F3F2A 161           D0=D0+ 2              If literal filespec, skip tLITRL!
F3F2D 3100          LC(2)  =tLITRL
F3F31 962           ?A=C   B              Is this a literal filespec?
F3F34 C2            GOYES  GETST1         Yes...exit, sSTK=0, skip tLITRL
F3F36 181           D0=D0- 2              No...undo D0=D0+ 2 done above
                *
                * This is not a literal, therefore must be a string expression
                *
F3F39 74C1          GOSUB  EXPEX+         Save status, evaluate the string
F3F3D 74D1          GOSUB  Restst         Restore status bits
F3F41 8F00  =GETST+ GOSBVL =REVPOP        Reverse it and pop it!
      000
                *
                * Now A[A] is the length, D1 points to the first byte!
                *
F3F48 850           ST=1   =sSTK          This is off the stack!
F3F4B 137           CD1EX
F3F4E D7            D=C    A              Save start of string in D[A]
F3F50 C2            C=C+A  A              Now C[A] points to string end
F3F52 8E00          GOSUBL =D1=AVE
      00
F3F58 145           DAT1=C A              ...write it out...
F3F5B DF            CDEX   A              ...put end in D[A], start in C[A]
F3F5D 135           D1=C                  ...put in D1...
F3F60 03    GETST1  RTNCC                 ...and return with all set up!
                ****************************************************************
                ****************************************************************
                **
                ** Name:       NXTCHR - Get next character from input
                **
                ** Category:   EXCUTL
                **
                ** Purpose:
                **       Get the next character from the input string
                **
```

```
866                ** Entry:
867                **      D1 points to next byte, if any
868                **      ST(sSTK) is status:  1--> Reading from stack
869                **                           0--> Reading from program memory
870                **      IF ST(sSTK)=1, D[A] is the end of the string
871                **
872                ** Exit:
873                **      P=0 if sSTK=0, P=(unchanged) if sSTK=1
874                **      If carry clear, A[B] is the next byte
875                **      If carry set, reached end of string
876                **        (If sSTK=0, A[B] is terminating character)
877                **
878                ** Calls:       None
879                **
880                ** Uses.......
881                **  Inclusive: A[B],D0,D1,P (D0 if sSTK=0, D1 if sSTK=1)
882                **
883                ** Stk lvls:   0
884                **
885                ** History:
886                **
887                **    Date      Programmer              Modification
888                **  --------   ----------    --------------------------------
889                **  11/12/82      NZ         Added documentation
890                **
891                ********************************************************************
892                ********************************************************************
893 F3F62 870   =NXTCHR ?ST=1  =sSTK
894 F3F65 71            GOYES  NXTCH1
895 F3F67 14A           A=DAT0 B
896 F3F6A 21            P=     1
897 F3F6C B04           A=A+1  P
898 F3F6F A0C           A=A-1  P
899 F3F72 20            P=     0
900 F3F74 400           RTNC
901 F3F77 161           D0=D0+ 2
902 F3F7A 03            RTNCC
903                *-
904                *-
905 F3F7C 14B   NXTCH1  A=DAT1 B
906 F3F7F 137           CD1EX
907 F3F82 8BF           ?C>=D  A
908 F3F85 20            GOYES  NXTCH2
909 F3F87 137   NXTCH2  CD1EX
910 F3F8A 400           RTNC
911 F3F8D 171           D1=D1+ 2
912 F3F90 03            RTNCC
913                ********************************************************************
914                ********************************************************************
915                **
916                ** Name:        LSTCHR - Unsupported entry point
917                **
918                ** Category:    LOCAL (EXCUTL)
919                **
920                ** Purpose:
```

```
                **      Inverse of nxtchr (Reverses the pointers)
                **
                ** Entry:
                **      Same as NXTCHR
                **
                ** Exit:
                **      D0 or D1 adjusted to the last character
                **      A[B] is the last character
                **      Carry set if D0/D1 NOT changed
                **
                ** Calls:      None
                **
                ** Uses.......
                **  Inclusive: A[B],D0,D1,P
                **
                ** Stk lvls:   0
                **
                ** Detail:
                **      NOTE!!! If reading from program memory AND NEXT char
                **      is a terminator, LSTCHR will NOT back up!
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------    --------------------------------
                **  11/12/82      NZ          Added documentation
                **
                ********************************************************************
                ********************************************************************
F3F92 870       =LSTCHR ?ST=1   =sSTK
F3F95 A1                GOYES   LSTCH1
F3F97 14A               A=DAT0  B
F3F9A 21                P=      1
F3F9C B04               A=A+1   P
F3F9F A0C               A=A-1   P
F3FA2 400               RTNC
F3FA5 181               D0=D0-  2
F3FA8 14A               A=DAT0  B
F3FAB 20                P=      0
F3FAD 03                RTNCC
                *-
                *-
F3FAF 137       LSTCH1  CD1EX
F3FB2 8BF               ?C>=D   A
F3FB5 20                GOYES   LSTCH2
F3FB7 137       LSTCH2  CD1EX
F3FBA 400               RTNC
F3FBD 1C1               D1=D1-  2
F3FC0 03                RTNCC
                ********************************************************************
                ********************************************************************
                **
                ** Name:        BAKCHR
                **
                ** Category:    EXCUTL
```

```
                        **
                        ** Purpose:
                        **      Unconditionally back up one character (undoes the
                        **      operation of NXTCHR, only IF a NXTCHR has been done)
                        **
                        ** Entry:
                        **      ST(=sSTK):
                        **        1: Reading from stack (@ D1)
                        **        0: Reading from memory (@ D0)
                        **
                        ** Exit:
                        **      D0/D1 adjusted according to sSTK
                        **      Carry clear
                        **
                        ** Calls:      None
                        **
                        ** Uses.......
                        **  Inclusive: D0,D1 (D0 if sSTK=0, D1 if sSTK=1)
                        **
                        ** Stk lvls:   0
                        **
                        ** Detail:
                        **      Allows backing up input stream one character if the
                        **      caller knows that there is a character before current
                        **      character
                        **
                        ** History:
                        **
                        **    Date     Programmer             Modification
                        **  --------   ----------    ---------------------------------
                        **  09/26/83      NZ         Updated documentation
                        **  11/12/82      NZ         Added documentation
                        **
                        ****************************************************************
                        ****************************************************************
F3FC2 870   =BAKCHR ?ST=1   =sSTK
F3FC5 70            GOYES   BAKCH1          String...back up D1
F3FC7 181           D0=D0- 2                Literal...back up D0
F3FCA 03            RTNCC
                        *-
                        *-
F3FCC 1C1   BAKCH1  D1=D1- 2                Back up D1
F3FCF 03            RTNCC
                        ****************************************************************
                        ****************************************************************
                        **
                        ** Name:       GETHEX - Evaluate literal expr, return hex value
                        **
                        ** Category:   GENUTL
                        **
                        ** Purpose:
                        **      Get the value of an expression in program memory
                        **
                        ** Entry:
                        **      D0 points to the expression in program memory
```

```
1031                **
1032                ** Exit:
1033                **      Carry clear: HEX value in A[3:0], A[4]=0, P=0
1034                **      Carry set: Error (P=error #)
1035                **
1036                ** Calls:      EXPEX+,FLTDH,AVM+16,RESTST
1037                **
1038                ** Uses.......
1039                **  Exclusive:     C,                          P
1040                **  Inclusive: A,B,C,D,R0,R1,R2,R3,R4,D0,D1,P,FUNCxx
1041                **
1042                ** Stk lvls:   5 (EXPEX+)
1043                **
1044                ** History:
1045                **
1046                **    Date      Programmer              Modification
1047                **  --------    ----------    ------------------------------
1048                **  03/16/83      NZ          Changed to EXPEX+, added RESTST
1049                **  11/12/82      NZ          Added documentation
1050                **
1051                ****************************************************************
1052                ****************************************************************
1053 F3FD1 7C21 =GETHEX GOSUB  EXPEX+         Save status, call EXPEXC
1054 F3FD5 7C31         GOSUB  Restst         Restore status
1055 F3FD9 75E0         GOSUB  AVM+16         pop it off the stack, reset AVMEME
1056 F3FDD 309          LCHEX  9
1057 F3FE0 98A          ?C>=A  P              Real number?
1058 F3FE3 60           GOYES  GETHE1         Yes!
1059                *
1060                * Not real...must be complex or string?
1061                *
1062 F3FE5 20           P=     =eNNUMR        Not real number!
1063 F3FE7 02           RTNSC
1064                *-
1065                *-
1066 F3FE9 8E00 GETHE1  GOSUBL =fLTDH         Convert to HEX number
          00
1067 F3FEF 5D0          GONC   GETHE3         Either <0 OR too big...error!
1068 F3FF2 24           P=     4              OK number...check MY range!
1069 F3FF4 90C          ?A#0   P
1070 F3FF7 60           GOYES  GETHE3         Positive, four or fewer digits
1071 F3FF9 20   GETHE2  P=     0              Reset P=0
1072 F3FFB 03           RTNCC
1073                *-
1074                *-
1075 F3FFD 20   GETHE3  P=     =eRANGE        Range error!
1076 F3FFF 02           RTNSC
1077                ****************************************************************
1078                ****************************************************************
1079                **
1080                ** Name:       GTYPRM - Get one-byte hex value from literal
1081                ** Name:       GTYPR+ - Clear status bits 11:0, GTYPRM
1082                ** Name:       GHEXBT - Pop number off stack, get hex byte value
1083                ** Name:       GHEXB+ - Use A[W] as value, convert to hex byte
1084                **
```

```
1085                ** Category:   EXCUTL
1086                **
1087                ** Purpose:
1088                **      Given DO pointing to a numeric expression in program
1089                **      memory, return the HEX value of the expression
1090                **
1091                ** Entry:
1092                **      ST(sSTK)=0: DO points to the expression
1093                **      ST(sSTK)=1: A[W] contains a floating number
1094                **
1095                ** Exit:
1096                **      If carry clear, B[B] is the HEX type, B[4:2]=0,P=0,
1097                **        C[B]=(DevTyp), C[XS]=0
1098                **      If carry set, error (P=type)
1099                **
1100                ** Calls:      EXPEX+,RESTST,AVM+16,FLTDH
1101                **
1102                ** Uses.......
1103                **  Exclusive: A,B,C,                          P
1104                **  Inclusive: A,B,C,D,R0,R1,R2,R3,R4,D0,D1,P,FUNCxx
1105                **
1106                ** Stk lvls:   5 (EXPEX+)
1107                **
1108                ** History:
1109                **
1110                **    Date      Programmer            Modification
1111                **  --------    ----------    ---------------------------------
1112                **  03/16/83       NZ         Changed to EXPEX+, added RESTST
1113                **  03/02/83       NZ         Added GTYPR+ entry point
1114                **  11/12/82       NZ         Added documentation
1115                **
1116                ****************************************************************
1117                ****************************************************************
1118 F4001 08    =GTYPR+ CLRST                   Clear all status bits
1119 F4003 20    =GTYPRM P=      0
1120 F4005 870         ?ST=1   =sSTK           Is expression in A[W] now?
1121 F4008 E0          GOYES   GTYPR0          Yes...skip EXPEX+
1122 F400A 73F0        GOSUB   EXPEX+          Expression execution
1123 F400E 7301        GOSUB   Reستst          Restore status
1124 F4012 7CA0 =GHEXBT GOSUB  AVM+16          Add 16 to AVMEME
1125 F4016      =GHEXB+
1126 F4016 309  GTYPR0  LCHEX   9
1127 F4019 986         ?C<A    P
1128 F401C C1          GOYES   GTYPRe          Not a floating number...error
1129 F401E 8E00        GOSUBL  =fLTDH          Convert to HEX
          00
1130 F4024 571         GONC    GTYPRr          Error!
1131 F4027 D1          B=0     A
1132 F4029 AEC         ABEX    B               Check if A[4:2] is zero
1133 F402C 8AC         ?A#0    A               Zero?
1134 F402F D0          GOYES   GTYPRr          No...range error!
1135                *
1136                * Now B[A] is the ID in HEX
1137                *
1138 F4031 3200        LC(3)   =DevTyp         This is a device TYPE!
```

```
          0
1139 F4036 01          RTN
1140           *-
1141           *-
1142 F4038 20  GTYPRe  P=     =eNNUMR
1143 F403A 02          RTNSC
1144           *-
1145           *-
1146 F403C 20  GTYPRr  P=     =eRANGE       Out of range!
1147 F403E 02          RTNSC
1148           ****************************************************************
1149           ****************************************************************
1150           **
1151           ** Name:       GADRRM - Get HPIL address from program memory
1152           ** Name:       GADRR+ - Get HPIL address from stack value
1153           **
1154           ** Category:   PILUTL
1155           **
1156           ** Purpose:
1157           **      Get an HPIL address from program memory
1158           **
1159           ** Entry:
1160           **      ST(sSTK)=0: D0 points to the expression in program mem
1161           **      ST(sSTK)=1: A[W] contains a floating number
1162           **
1163           ** Exit:
1164           **      Carry clear: C[X] is the HPIL address, P=0
1165           **      Carry set: Error (P is error #)
1166           **
1167           ** Calls:      EXPEX+,RESTST,AVM+16,GHEXB+
1168           **
1169           ** Uses.......
1170           **  Exclusive: A,B,C,D,                        P
1171           **  Inclusive: A,B,C,D,R0,R1,R2,R3,R4,D0,D1,P,FUNCxx
1172           **
1173           ** Stk lvls:   5 (EXPEX+)
1174           **
1175           ** History:
1176           **
1177           **    Date     Programmer             Modification
1178           **  --------   ----------    --------------------------------
1179           **  07/13/83      NZ         Added check for primary addr=0
1180           **  03/16/83      NZ         Changed to EXPEX+,added RESTST
1181           **  11/12/82      NZ         Added documentation
1182           **
1183           ****************************************************************
1184           ****************************************************************
1185 F4040 20 =GADRRM P=     0
1186 F4042 870         ?ST=1  =sSTK         Is expression already in A[W]?
1187 F4045 E0          GOYES  GADRR0        Yes...skip EXPEX+
1188 F4047 76B0        GOSUB  EXPEX+        EXPression EXCution
1189 F404B 76C0        GOSUB  Restst        Restore status bits
1190 F404F 7F60 =GADRR+ GOSUB  AVM+16       Skip the item
1191 F4053 AF6  GADRR0  C=A    W
1192 F4056 AF7         D=C    W             Save the expression in D
```

```
1193 F4059 79BF          GOSUB  GHEXB+        Get HEX byte (Primary address)
1194 F405D 400           RTNC                 Error...range error
1195 F4060 D9            C=B    A
1196 F4062 AFF           CDEX   W             Save IP in D[A], get back expr
1197 F4065 AFA           A=C    W             Put expression in A[W]
1198 F4068 94C           ?A#0   S
1199 F406B 35            GOYES  GADDRr        Negative!!
1200 F406D 3260          LC(3)  6             If exp >6 (or negative), error!
          0
1201 F4072 9B6           ?A>C   X
1202 F4075 94            GOYES  GADDRr        Error (range)
1203 F4077 A86           C=A    P
1204 F407A B8E           C=-C-1 P
1205 F407D 80D0          P=C    0             Now P-->First fractional digit+2
1206 F4081 BD0  GADRR1   ASL    M
1207 F4084 0C            P=P+1
1208 F4086 5AF           GONC   GADRR1        Go if not done yet...
1209             *
1210             * Now the mantissa is properly adjusted to the fractional part
1211             * (The mantissa has the original integer part removed)
1212             *
1213 F4089 D0            A=0    A
1214 F408B BF0           ASL    W             Normalize the number!
1215 F408E 948           ?A=0   S
1216 F4091 70            GOYES  GADRR2        Now is normalized!
1217 F4093 BF4           ASR    W
1218 F4096 E4            A=A+1  A             Exponent=1 means use 2 digits
1219 F4098 7A7F GADRR2   GOSUB  GHEXB+
1220 F409C 400           RTNC                 GHEXB+ sets HEX mode
1221             *
1222             * Now B[B] is secondary address, D[B] is primary address
1223             *
1224 F409F 31F1          LC(2)  31            Check range of secondary address
1225 F40A3 9E1           ?B>C   B             Is it legal range? [0,31]
1226 F40A6 81            GOYES  GADDRr        No!!!
1227 F40A8 9EB           ?D>=C  B
1228 F40AB 31            GOYES  GADDRr        Bad primary range!
1229 F40AD 96B           ?D=0   B
1230 F40B0 E0            GOYES  GADDRr        Primary must be >0!
1231 F40B2 D9            C=B    A
1232 F40B4 F2            CSL    A             Shift the secondary address left
1233 F40B6 C6            C=C+C  A               5 bits...then OR with D[X]
1234 F40B8 0EFF          C=C!D  A             Now address is in C[X]
1235 F40BC 03            RTNCC
1236             *-
1237             *-
1238 F40BE 20   GADDRr   P=     =eRANGE
1239 F40C0 02            RTNSC
1240             ********************************************************************
1241             ********************************************************************
1242             **
1243             ** Name:       AVM+16 - Pop a numeric value from AVMEME
1244             **
1245             ** Category:   PTRUTL
1246             **
```

```
1247                ** Purpose:
1248                **      Add 16 to AVMEME (to skip a numeric expression) and
1249                **      read in the value at the old D1
1250                **
1251                ** Entry:
1252                **      AVMEME stack has a numeric item
1253                **
1254                ** Exit:
1255                **      A[W] contains the old stack data item
1256                **      D1 points to old (=AVMEME)
1257                **      C[A] is NEW =AVMEME
1258                **      Carry unchanged
1259                **
1260                ** Calls:       D1=AVE
1261                **
1262                ** Uses.......
1263                **  Inclusive: A[W],C[A],C[S],D1
1264                **
1265                ** Stk lvls:   1 (D1=AVE)
1266                **
1267                ** NOTE: Preserves carry!!!!
1268                **
1269                ** History:
1270                **
1271                **    Date      Programmer              Modification
1272                **  --------    ----------    ---------------------------------
1273                **  07/13/83       NZ         Added read of A[W]
1274                **  11/12/82       NZ         Added documentation
1275                **
1276                ******************************************************************
1277                ******************************************************************
1278 F40C2 AC2    =AVM+16 C=0     S               Save carry status in C[S]
1279 F40C5 450            GOC     AVM++
1280 F40C8 B46            C=C+1   S
1281 F40CB 8E00 AVM++     GOSUBL  =D1=AVE
          00
1282 F40D1 147            C=DAT1  A
1283 F40D4 137            CD1EX
1284 F40D7 17F            D1=D1+  16
1285 F40DA 137            CD1EX
1286 F40DD 145            DAT1=C  A
1287 F40E0 135            D1=C                    Leave D1-->AVMEME-16
1288 F40E3 1CF            D1=D1-  16
1289 F40E6 1537           A=DAT1  W               Read in the value to A[W]
1290 F40EA A4E            C=C-1   S               Sets carry if zero, else clears
1291 F40ED 01             RTN
1292                *-
1293                *-
1294 F40EF 816    Csrc5   CSRC
1295 F40F2 8C00 Csrc4     GOLONG  =CSRC4
          00
1296                *-
1297                *-
1298 F40F8 812    Cslc5   CSLC
1299 F40FB 8C00 Cslc4     GOLONG  =CSLC4
```

```
                00
1300                    *-
1301                    *-
1302 F4101 8E00 =EXPEX+ GOSUBL =SAVEST
                00
1303 F4107 8D00 =eXPEXC GOVLNG =EXPEXC
                000
1304                    *-
1305                    *-
1306 F410E 8D00 =i/OFND GOVLNG =I/OFND
                000
1307                    *-
1308                    *-
1309 F4115 8C00 Restst  GOLONG =RESTST
                00
1310                    ********************************************************************
1311                    ********************************************************************
1312                    **
1313                    ** Name:        CHKAIO - Check if device is an ASSIGN WORD
1314                    **
1315                    ** Category:    PILUTL
1316                    **
1317                    ** Purpose:
1318                    **      Check if a string is an ASSIGN WORD (if so, return
1319                    **      its value)
1320                    **
1321                    ** Entry:
1322                    **      B contains a string (B[B] is the first character, any
1323                    **      unused characters are #00)
1324                    **
1325                    ** Exit:
1326                    **      P=0
1327                    **      Carry set if buffer not found or not an ASSIGN WORD
1328                    **      Carry clear if found...address in C[X]
1329                    **
1330                    ** Calls:       CSLC5,ASRC5,I/OFND
1331                    **
1332                    ** Uses.......
1333                    **  Exclusive: A[W],C[W],P
1334                    **  Inclusive: A[W],C[W],P
1335                    **
1336                    ** Stk lvls:    1 (I/OFND)(CSLC5)(ASRC5)
1337                    **
1338                    ** History:
1339                    **
1340                    **    Date      Programmer              Modification
1341                    **  --------    ----------    --------------------------------
1342                    **  11/12/82      NZ          Added documentation
1343                    **
1344                    ********************************************************************
1345                    ********************************************************************
1346 F411B 137  =CHKAIO CD1EX                 Save D1 from I/OFND in C[9:5]
1347 F411E 76DF         GOSUB   Cslc5
1348 F4122 20           P=      0
1349 F4124 3200         LC(3)   =bPILAI       ASSIGN IO buffer ID
```

```
          0
1350 F4129 71EF         GOSUB  1/OFND        I/O FiND
1351 F412D AFA          A=C    W             Save D1 in A[9:5]
1352 F4130 AF2          C=0    W
1353 F4133 04           SETHEX
1354 F4135 490          GOC    CHKAIO        Found...
1355 F4138 2F           P=     15
1356 F413A 0C           P=P+1                Set carry, P=0
1357 F413C 4F1          GOC    CHKAI3        Go always (not found...restore D1)
1358              *-
1359              *-
1360              *
1361              * D1--> Table of assignments (length of 30 entries*4 nibbles)
1362              *
1363 F413F 20   CHKAIO  P=     0
1364 F4141 D0           A=0    A             Address counter
1365 F4143 E4   CHKAI1  A=A+1  A             Increment A[B]
1366 F4145 31F1         LC(2)  31            Check if done
1367 F4149 9EE          ?A>=C  B
1368 F414C E0           GOYES  CHKAI2        Done...not found!
1369 F414E 15F3         C=DAT1 4
1370 F4152 173          D1=D1+ 4
1371 F4155 975          ?B#C   W
1372 F4158 BE           GOYES  CHKAI1        Not a match.
1373              *
1374              * If carry clear, found it; else not found
1375              *
1376 F415A D6   CHKAI2  C=A    A             Copy address to C[X]
1377 F415C 8E00 CHKAI3  GOSUBL =ASRC5        Not found!
          00
1378 F4162 131          D1=A                 Restore D1
1379 F4165 01           RTN                  Return, carry unchanged
1380              ****************************************************************
1381              ****************************************************************
1382              **
1383              ** Name:       ROMTYP - Check if device is a RESERVED WORD
1384              **
1385              ** Category:   PILUTL
1386              **
1387              ** Purpose:
1388              **      Check if the string in B[W] is a RESERVED WORD; if so,
1389              **      return the value that corresponds to that word
1390              **
1391              ** Entry:
1392              **      B contains the string (B[B] is the first character)
1393              **
1394              ** Exit:
1395              **      P=0
1396              **      Carry clear: B[B] is the device type; B[XS]=0
1397              **      Carry set: not found
1398              **
1399              ** Calls:      None
1400              **
1401              ** Uses.......
1402              **  Inclusive: B[A],C[W],P  (B[A] only if found)
```

```
1403                **
1404                ** Stk lvls:   1 (Internal call)(internal push)
1405                **
1406                ** History:
1407                **
1408                **    Date     Programmer            Modification
1409                **  --------   ----------   ---------------------------------
1410                **  09/26/83      NZ         Updated documentation
1411                **  11/12/82      NZ         Added documentation
1412                **
1413                ****************************************************************
1414                ****************************************************************
1415 F4167 72A0 =ROMTYP GOSUB  ROMTY1
1416                *
1417                * TABLE!!!
1418                *
1419                *
1420                * The table entry structure is:
1421                *       1 nibble: length of name minus 1, in nibbles (n-1)
1422                *       n nibbles: name (Bytes in order!)
1423                *       2 nibbles: device type
1424                *
1425                * The table consists of entries terminated by length nibble=0
1426                *
1427 F416B 7            NIBHEX 7              Length of "TAPE"
1428 F416C 4514         NIBASC \TAPE\         TAPE:TYPE=10
            0554
1429 F4174 01           NIBHEX 01
1430 F4176 D            NIBHEX D              Length of "MASSMEM"
1431 F4177 D414         NIBASC \MASSMEM\      MASSMEM:TYPE=1F (MASS MEM. CLASS)
            3535
            D454
            D4
1432 F4185 F1           NIBHEX F1
1433 F4187 D            NIBHEX D              Length of "PRINTER"
1434 F4188 0525         NIBASC \PRINTER\      PRINTER:TYPE=2F (PRINTER CLASS)
            94E4
            4554
            25
1435 F4196 F2           NIBHEX F2
1436 F4198 D            NIBHEX D              Length of "DISPLAY"
1437 F4199 4494         NIBASC \DISPLAY\      DISPLAY:TYPE=3F (DISPLAY CLASS)
            3505
            C414
            95
1438 F41A7 F3           NIBHEX F3
1439 F41A9 7            NIBHEX 7              Length of "GPIO"
1440 F41AA 7405         NIBASC \GPIO\         GPIO:TYPE=40
            94F4
1441 F41B2 04           NIBHEX 04
1442 F41B4 9            NIBHEX 9              Length of "MODEM"
1443 F41B5 D4F4         NIBASC \MODEM\        MODEM:TYPE=41
            4454
            D4
1444 F41BF 14           NIBHEX 14
```

```
1445 F41C1 9           NIBHEX 9             Length of "RS232"
1446 F41C2 2535        NIBASC \RS232\       RS232:TYPE=42
           2333
           23
1447 F41CC 24          NIBHEX 24
1448 F41CE 7           NIBHEX 7             Length of "HPIB"
1449 F41CF 8405        NIBASC \HPIB\        HPIB:TYPE=43
           9424
1450 F41D7 34          NIBHEX 34
1451 F41D9 D           NIBHEX D             Length of "INTRFCE"
1452 F41DA 94E4        NIBASC \INTRFCE\     INTRFCE:TYPE=4F
           4525
           6434
           54
1453 F41E8 F4          NIBHEX F4
1454 F41EA D           NIBHEX D             Length of "INSTRMT"
1455 F41EB 94E4        NIBASC \INSTRMT\     INSTRMT:TYPE=5F (INSTRMT CLASS)
           3545
           25D4
           45
1456 F41F9 F5          NIBHEX F5
1457 F41FB D           NIBHEX D             Length of "GRAPHIC"
1458 F41FC 7425        NIBASC \GRAPHIC\     GRAPHIC:TYPE=6F (GRAPHIC I/O)
           1405
           8494
           34
1459 F420A F6          NIBHEX F6
1460           * END OF TABLE INDICATOR...NULL
1461 F420C 0           NIBHEX 0
1462           *
1463           * END OF TABLE!
1464           *
1465 F420D 07  ROMTY1  C=RSTK               Get pointer to table from stack..
1466 F420F 137         CD1EX                ..Put it in D1, put D1 in C[A]..
1467 F4212 06          RSTK=C               ..and save D1 value on the stack!
1468           *
1469           * Loop to process names...
1470           *
1471 F4214 AF2 ROMTY2  C=0    W
1472 F4217 14F         C=DAT1 B             Read length of the device word
1473 F421A 170         D1=D1+ 1
1474 F421D 80D0        P=C    0             Copy length into P
1475 F4221 890         ?P=    0             END OF TABLE??
1476 F4224 12          GOYES  ROMTY3        Yes...restore D1, P; carry set!
1477           *
1478           * Have a non-zero length now!
1479           *
1480 F4226 1571        C=DAT1 WP            Read the device word...
1481           *
1482 F422A 171         D1=D1+ 2             Increment D1 by the length +2
1483 F422D 137         CD1EX
1484 F4230 809         C+P+1                If match, back off the +2!
1485 F4233 137         CD1EX
1486           *
1487           * Now C[W] is the device word, zero-filled (if blank-filled is
```

```
1488            * desired, change the C=0 W above to a LCASC \       \)
1489            *
1490 F4236 975          ?B#C   W
1491 F4239 BD           GOYES  ROMTY2        Not matched!
1492            *
1493            * This is a match...continue!
1494            *
1495 F423B 1C1          D1=D1- 2             Point to device type byte...
1496            *
1497            * (Carry is clear from the statement above)
1498            *
1499 F423E D2           C=0    A             Clear C[XS]...
1500 F4240 14F          C=DAT1 B             Read device type!
1501 F4243 D5           B=C    A             Copy C[X] to B[X]
1502            *
1503            * Common return point!
1504            *
1505 F4245 07   ROMTY3  C=RSTK
1506 F4247 135          D1=C                 Restore D1...
1507 F424A 20           P=     0
1508 F424C D2           C=0    A
1509 F424E 3100         LC(2)  =DevTyp       Device type
1510 F4252 01           RTN                  ...and return, carry unchanged!
1511            ********************************************************************
1512            ********************************************************************
1513            **
1514            ** Name:       RDINFO - Read device info from SAVSTK + POLL
1515            **
1516            ** Category:   SAVSTK
1517            **
1518            ** Purpose:
1519            **      Read information from the SAVSTK, given one POLL level
1520            **      in front of the data
1521            **
1522            ** Entry:
1523            **      ST(=sDEST) is source/destination selector
1524            **
1525            ** Exit:
1526            **      P=0
1527            **      A[W] is first 8 chars
1528            **      R0 is last 2 chars
1529            **      D[A] is device
1530            **
1531            ** Calls:      None
1532            **
1533            ** Uses.......
1534            **  Inclusive: A[W],C[A],D[A],R0,D1,P
1535            **
1536            ** Stk lvls:   0
1537            **
1538            ** NOTE: This is similar to the mainframe routine by the same
1539            **       name except for the first few lines which skip the
1540            **       POLL save area
1541            **
1542            ** History:
```

```
1543                **
1544                **    Date      Programmer            Modification
1545                **  --------    ----------    --------------------------------
1546                **  11/12/82       NZ         Added documentation
1547                **
1548                ****************************************************************
1549                ****************************************************************
1550 F4254 1F00 =RDINFO D1=(5) =SAVSTK
             000
1551 F425B 143          A=DAT1 A
1552 F425E 20           P=     0
1553 F4260 D2           C=0    A
1554 F4262 3100         LC(2)  =lPOLSV       Length of POLL save area
1555 F4266 EA           A=A-C  A
1556 F4268 131          D1=A                 D1-->device save area
1557 F426B 1C0          D1=D1- (=lDEVC)+4    Length of device +2 chars of name
1558 F426E 1CF          D1=D1- 16            Length of 8 chars of name
1559 F4271 860          ?ST=0  =sDEST
1560 F4274 80           GOYES  RDIN10
1561 F4276 1C0          D1=D1- (=lDEVC)+4
1562 F4279 1CF          D1=D1- 16            Skip source info
1563 F427C 1537 RDIN10  A=DAT1 W             First 8 chars
1564 F4280 17F          D1=D1+ 16            Move past them
1565 F4283 147          C=DAT1 A             Last 2 chars
1566 F4286 108          R0=C                 -->R0
1567 F4289 173          D1=D1+ 4             Skip last 2 chars
1568 F428C 147          C=DAT1 A             Device info
1569 F428F D7           D=C    A             -->D
1570 F4291 03           RTNCC
1571 F4293              END
```

```
 ASRC5   Ext                    -  1377
 AVM++   Abs  999627 #F40CB -  1281  1279
=AVM+16  Abs  999618 #F40C2 -  1278  1055  1124  1190
 BAKCH1  Abs  999372 #F3FCC -  1017  1012
=BAKCHR  Abs  999362 #F3FC2 -  1011
 CHKAI0  Abs  999743 #F413F -  1363  1354
 CHKAI1  Abs  999747 #F4143 -  1365  1372
 CHKAI2  Abs  999770 #F415A -  1376  1368
 CHKAI3  Abs  999772 #F415C -  1377  1357
=CHKAI0  Abs  999707 #F411B -  1346
 CHKAS!  Abs  998673 #F3D11 -   346   345
 CHKAS#  Abs  998666 #F3D0A -   343   336
 CHKAS0  Abs  998677 #F3D15 -   350   339
 CHKAS1  Abs  998702 #F3D2E -   368   361
 CHKAS2  Abs  998754 #F3D62 -   404   383
 CHKAS3  Abs  998801 #F3D91 -   439   437
 CHKAS4  Abs  998804 #F3D94 -   441
 CHKAS9  Abs  998815 #F3D9F -   453   362
=CHKASN  Abs  998636 #F3CEC -   303
 CHKASs  Abs  998827 #F3DAB -   461   412
 CHKASx  Abs  998813 #F3D9D -   447   413
 CNFFND  Ext                    -   210
 CSLC4   Ext                    -   589  1299
 CSLC9   Ext                    -   690
 CSRC3   Ext                    -   713
 CSRC4   Ext                    -  1295
 Cslc4   Abs  999675 #F40FB -  1299   724
 Cslc5   Abs  999672 #F40F8 -  1298   204   563  1347
 Csrc4   Abs  999666 #F40F2 -  1295   584   675   688
 Csrc5   Abs  999663 #F40EF -  1294   241   711
 D1=AVE  Ext                    -   851  1281
 DSPSET  Ext                    -   193
 DevID   Ext                    -   435
 DevTyp  Ext                    -   399  1138  1509
 DispOK  Ext                    -   196
 DsNull  Ext                    -   568
 ERRORX  Ext                    -   733
=EXPEX+  Abs  999681 #F4101 -  1302   841  1053  1122  1188
 EXPEXC  Ext                    -  1303
=FNDMB+  Abs  998460 #F3C3C -   168
=FNDMB-  Abs  998464 #F3C40 -   172
 FNDMB.  Abs  998484 #F3C54 -   185   184
 FNDMB1  Abs  998558 #F3C9E -   217   229
 FNDMB2  Abs  998572 #F3CAC -   225   251
 FNDMB3  Abs  998607 #F3CCF -   246   221
 FNDMB9  Abs  998593 #F3CC1 -   239   187   267
=FNDMBD  Abs  998495 #F3C5F -   193
 FNDMBE  Abs  998587 #F3CBB -   233   211   218
=FNDMBX  Abs  998517 #F3C75 -   202
 GADDRr  Abs  999614 #F40BE -  1238  1199  1202  1226  1228  1230
=GADRR+  Abs  999503 #F404F -  1190
 GADRR0  Abs  999507 #F4053 -  1191  1187
 GADRR1  Abs  999553 #F4081 -  1206  1208
 GADRR2  Abs  999576 #F4098 -  1219  1216
=GADRRM  Abs  999488 #F4040 -  1185
```

```
 GETHE1   Abs  999401 #F3FE9 -  1066  1058
 GETHE2   Abs  999417 #F3FF9 -  1071
 GETHE3   Abs  999421 #F3FFD -  1075  1067  1070
=GETHEX   Abs  999377 #F3FD1 -  1053
=GETMBX   Abs  998391 #F3BF7 -    47
=GETST+   Abs  999233 #F3F41 -   843
 GETST1   Abs  999264 #F3F60 -   855   829   836
=GETSTR   Abs  999193 #F3F19 -   821
=GHEXB+   Abs  999446 #F4016 -  1125  1193  1219
=GHEXBT   Abs  999442 #F4012 -  1124
=GTYPR+   Abs  999425 #F4001 -  1118
 GTYPRO   Abs  999446 #F4016 -  1126  1121
=GTYPRM   Abs  999427 #F4003 -  1119
 GTYPRe   Abs  999480 #F4038 -  1142  1128
 GTYPRr   Abs  999484 #F403C -  1146  1130  1134
 I/OALL   Ext                -   704
 I/OFND   Ext                -  1306
 I/OFSC   Ext                -   695
 I/odal   Ext                -   676
 IS-DSP   Ext                -   767
 LOOPST   Ext                -   176
 LSTCH1   Abs  999343 #F3FAF -   963   951
 LSTCH2   Abs  999351 #F3FB7 -   966   965
=LSTCHR   Abs  999314 #F3F92 -   950
 MBOX^    Ext                -    47   258
=NORAMe   Abs  999145 #F3EE9 -   732   696   705
 NXTCH1   Abs  999292 #F3F7C -   905   894
 NXTCH2   Abs  999303 #F3F87 -   909   908
=NXTCHR   Abs  999266 #F3F62 -   893
 Offed    Ext                -   183
 PILCNs   Abs  999167 #F3EFF -   773   768
 PILCns   Abs  999191 #F3F17 -   781   776
 RDIN10   Abs 1000060 #F427C -  1563  1560
=RDINFO   Abs 1000020 #F4254 -  1550
=RESTOR   Abs  999153 #F3EF1 -   767
 RESTST   Ext                -  1309
 REVPOP   Ext                -   843
 ROMTY1   Abs  999949 #F420D -  1465  1415
 ROMTY2   Abs  999956 #F4214 -  1471  1491
 ROMTY3   Abs 1000005 #F4245 -  1505  1476
=ROMTYP   Abs  999783 #F4167 -  1415
 Restst   Abs  999701 #F4115 -  1309   842  1054  1123  1189
 SAVEİ-   Abs  999012 #F3E64 -   673   672
 SAVEI0   Abs  998992 #F3E50 -   664   729
 SAVEI1   Abs  999031 #F3E77 -   681   660
=SAVEIT   Abs  998987 #F3E4B -   659
 SAVEST   Ext                -  1302
 SAVSTK   Ext                -  1550
=SETLP    Abs  998418 #F3C12 -    94   168
 SETLP1   Abs  998441 #F3C29 -   111   103
 SETLP2   Abs  998448 #F3C30 -   114   105
=SETUP    Abs  998856 #F3DC8 -   514
 SETUP,   Abs  998912 #F3E00 -   573   571
 SETUP0   Abs  998885 #F3DE5 -   563   553
 SETUP1   Abs  998917 #F3E05 -   577   555
```

```
 SETUP2  Abs  998947 #F3E23 -   593   557   559
 SETUPx  Abs  998894 #F3DEE -   566   590
 SngDev  Ext               -   382   671   726
 VolLbl  Ext               -   438
 bPILAI  Ext               -  1349
 eNMBOX  Ext               -   235
 eNNUMR  Ext               -  1062  1142
 eNORAM  Ext               -   732
 eOFFED  Ext               -   182
 eRANGE  Ext               -  1075  1146  1238
=eXPEXC  Abs  999687 #F4107 -  1303
 fLIDH   Ext               -  1066  1129
=i/OFND  Abs  999694 #F410E -  1306   472  1350
 lDEVC   Ext               -  1557  1561
 lPOLSV  Ext               -  1554
 sDEST   Ext               -  1559
 sSTK    Ext               -   821   847   893   950  1011  1120  1186
 tCOLON  Ext               -   827
 tLITRL  Ext               -   834
```

Input Parameters

  Source file name is NZ&BUT::MS

  Listing file name is NZ/BUT:TI:ML::-1

  Object file name is NZ%BUT:TI:MS::-1

```
                                    111111
                          0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1                *
 2                *       N   N  ZZZZZ   &      CCC     A     SSS
 3                *       N   N      Z  & &    C   C   A A   S   S
 4                *       NN  N     Z   & &    C      A   A  S
 5                *       N N N    Z     &     C      A   A   SSS
 6                *       N  NN   Z     & & &  C      AAAAA      S
 7                *       N   N  Z      &  &   C   C  A   A  S   S
 8                *       N   N  ZZZZZ   && &   CCC   A   A   SSS
 9                *
10                *
11                        TITLE  CASSETTE ROUTINES<831221.1632>
12 F4293                  ABS    #F4293        TI%HP6 address (fixed)
13                ****************************************************************
14                ****************************************************************
15                **
16                ** Name:       TSTAT,TSTATA - Check the drive status
17                **
18                ** Category:   PILUTL
19                **
20                ** Purpose:
21                **      Check status of mass storage device
22                **
23                ** Entry:
24                **      D[X] contains the address of the drive
25                **      D0 points to the mailbox
26                **
27                ** Exit:
28                **      Carry clear:
29                **        Drive is addressed as a talker
30                **        Status in C[B]
31                **      Carry set:
32                **        Error (P, C[0] are error code)
33                **
34                ** Calls:      YTML,PUTE,GETD (YTML only for TSTAT)
35                **
36                ** Uses.......
37                **  Exclusive: C[W],P
38                **  Inclusive: C[W],P,ST[3:0]
39                **
40                ** Stk lvls:   2 (YTML;PUTC)(GETD;GET)
41                **
42                ** History:
43                **
44                **    Date     Programmer             Modification
45                **  --------   ----------    ------------------------------
46                **  11/19/82      NZ         Added documentation
47                **
48                ****************************************************************
49                ****************************************************************
50 F4293 7DA5  =TSTAT   GOSUB  Ytml
51 F4297 400           RTNC                  Error
52 F429A 20    =TSTATA P=     0
53 F429C 3500          LC(6)  (=MSST)+1     Send status, limit=1
         0000
54 F42A4 8E00          GOSUBL =PUTE
```

```
          00
    55 F42AA 400          RTNC                    Error
    56 F42AD 7D85  TSTAT1 GOSUB  Getd
    57 F42B1 400          RTNC                    RTNSC if not data frame
    58 F42B4 80D1         P=C    1
    59 F42B8 880          ?P#    0                Is it either BUSY or Error?
    60 F42BB 40           GOYES  TSTAT2           Yes...check which!
    61 F42BD 03           RTNCC                   No...all OK
    62             *-
    63             *-
    64 F42BF 891   TSTAT2 ?P=    1                Is it an error?
    65 F42C2 00           RTNYES                  Yes...RTNSC
    66 F42C4 55D          GONC   TSTATA           No...must be busy...try again
    67             ****************************************************************
    68             ****************************************************************
    69             **
    70             ** Name:       SEEKA - Seek a record (record # in A[3:0])
    71             ** Name:       SEEKB - Seek record (drive=listener,me=talker)
    72             **
    73             ** Category:   PILUTL
    74             **
    75             ** Purpose:
    76             **      Seek to the specified record
    77             **
    78             ** Entry:
    79             **      SEEKA: Desired record # is in A[3:0]
    80             **      SEEKB: Desired record # is in A[3:0], drive is talker,
    81             **             I am listener
    82             **      Drive address in D[X]
    83             **      DO points to the mailbox
    84             **
    85             ** Exit:
    86             **      Carry clear:
    87             **        Drive is talker, I am listener, P=0
    88             **      Carry set:
    89             **        Error (P,C[0] are error code)
    90             **
    91             ** Calls:      MTYL,DDL,PUTD,<TSTAT>
    92             **
    93             ** Uses.......
    94             **  Exclusive: C[W],P
    95             **  Inclusive: C[W],P,ST[3:0]
    96             **
    97             ** Stk lvls:   2 (MTYL) <TSTAT>
    98             **
    99             ** History:
   100             **
   101             **    Date     Programmer              Modification
   102             **  --------   ----------    --------------------------------
   103             **  11/19/82      NZ         Added documentation
   104             **
   105             ****************************************************************
   106             ****************************************************************
   107 F42C7 72B7 =SEEKA  GOSUB  Mtyl
   108 F42CB 400          RTNC                    Error
```

```
F42CE 20    =SEEKB  P=      =Seek
F42D0 7367          GOSUB   Ddl
F42D4 400           RTNC                    Error
F42D7 D6            C=A     A               Get track # first
F42D9 F6            CSR     A
F42DB F6            CSR     A
F42DD 7097          GOSUB   Putd            Send track number
F42E1 400           RTNC                    Error
F42E4 D6            C=A     A               Now get record # on track
F42E6 7787          GOSUB   Putd            Send record number
F42EA 400           RTNC                    Error
                *
                * Following can be packed to GONC if needed
                *
F42ED 65AF          GOTO    TSTAT           Check status and exit
                ****************************************************************
                ****************************************************************
                **
                ** Name:        CHKMAS - Check if D[X] is mass storage device
                **
                ** Category:    PILUTL
                **
                ** Purpose:
                **      Check if a device (at D[X]) is mass storage
                **
                ** Entry:
                **      D[X] is device address
                **      D0 points to the mailbox
                **
                ** Exit:
                **      Carry clear:
                **        Device is mass storage (Acc ID=#10), P=0
                **      Carry set:
                **        Not mass storage OR loop error
                **        (P, C[0] are error code - if P= =ePIL, C[0]=eDTYPE,
                **        than C[1] is device class, A[B] is full Acc ID)
                **
                ** Calls:       GTYPE
                **
                ** Uses.......
                **  Exclusive:      C[W],P
                **  Inclusive: A[A],C[W],P,ST[3:0]
                **
                ** Stk lvls:    3 (GTYPE)
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------    --------------------------------
                **  05/25/83      NZ          Rewrote again to save code, added
                **                            exit condition for C[1] (device
                **                            class)
                **  02/16/83      NZ          Rewrote to not use mQSTAT, which
                **                            was removed from Diamond
                **                            (Added A[A] register usage)
```

```
164                **                              (Added 2 stack levels)
165                **  11/19/82      NZ            Added documentation
166                **
167                ****************************************************************
168                ****************************************************************
169 F42F1 8E00 =CHKMAS GOSUBL =GTYPE               Get the acc ID of the device in A
          00
170 F42F7 400          RTNC                       (Error)
171 F42FA 3101         LCHEX  10                  Check if Acc ID=16
172 F42FE 966          ?A#C   B
173 F4301 40           GOYES  CHKMAe              Not Acc ID=16
174 F4303 03   Rtncc   RTNCC
175                *-
176                *-
177 F4305 D6   CHKMAe  C=A    A                   Copy accessory ID to C[B] first
178 F4307 300          LC(1)  =eDTYPE             Device type error
179 F430A 20           P=     =ePIL
180 F430C 02           RTNSC
181                ****************************************************************
182                ****************************************************************
183                **
184                ** Name:       CHKBIT - Check if device indicates Acc ID=16
185                **
186                ** Category:   LOCAL
187                **
188                ** Purpose:
189                **      Check if bit "4" of D[3] is set or clear
190                **
191                ** Entry:
192                **      D[3:0] is device spec from file spec execute
193                **
194                ** Exit:
195                **      Carry set if bit is set (Acc ID=16 device)
196                **
197                ** Calls:      None
198                **
199                ** Uses.......
200                **  Inclusive: C[A]
201                **
202                ** Stk lvls:   0
203                **
204                ** History:
205                **
206                **    Date     Programmer             Modification
207                **  --------   ----------    ---------------------------------
208                **  05/12/83      NZ         Wrote routine and documentation
209                **
210                ****************************************************************
211                ****************************************************************
212 F430E DB   =CHKBIT C=D    A                   Copy to C[A] for checking
213 F4310 F2           CSL    A
214 F4312 C6          C=C+C  A
215 F4314 C6          C=C+C  A                   Check the desired bit
216 F4316 01           RTN                        Carry set iff bit set
217                ****************************************************************
```

```
                ****************************************************************
                **
                ** Name:       CLEARN - Clear a record on device (send zeroes)
                ** Name:       CLLOOP - Send 0's to a device (A[A] is count)
                **
                ** Category:   PILI/O
                **
                ** Purpose:
                **      Clear a record (output zeroes to a specific record)
                **
                ** Entry:
                **      D[X] contains the address of the drive
                **      Diamond is talker, drive is listener
                **      Record number in A[3:0]
                **      DO points to the mailbox
                **
                ** Exit:
                **      Carry clear:
                **        Successful (P=0)
                **      Carry set:
                **        Error (P, C[0] are error code)
                **
                ** Calls:      <SENDIT>
                **
                ** Uses.......
                **  Exclusive: A[A],B[W],     P
                **  Inclusive: A[A],B[W],C[W],P,ST[3:0]
                **
                ** Stk lvls:   1 <SENDIT>
                **
                ** History:
                **
                **    Date     Programmer              Modification
                **  --------   ----------    ---------------------------------
                **  03/22/83      NZ         Removed CLEARR entry point
                **  11/19/82      NZ         Added documentation
                **
                ****************************************************************
                ****************************************************************
F4318 D0        =CLEARN A=0     A
F431A B24               A=A+1   XS              Set A[A]<--#00100 (256)
F431D AF1       =CLLOOP B=0     W               A[A] is the # of bytes to clear
F4320 8C00              GOLONG  =SENDIT         Send all zeroes!
      00
                ****************************************************************
                ****************************************************************
                **
                ** Name:       FORMAT - Format medium in specified drive
                **
                ** Category:   EXCUTL
                **
                ** Purpose:
                **      Format medium in specified drive (initialize it)
                **
                ** Entry:
```

```
                **      R0 contains vol label ([11:0]), # of entries ([15:12])
                **      Drive address is in D[X]
                **      D0 points to the mailbox
                **
                ** Exit:
                **      Carry clear:
                **        P=0, drive is rewinding (successful formatting)
                **      Carry set:
                **        Error (P, C[0] are error code)
                **
                ** Calls:      DDL,DDT,READI3,WRITIT,PRMSGA,CLLOOP,CLEARN,
                **             MTYL,YTML,TSTAT,SEEKA,PUTALR,PUTDX,PUTD,PUTE,
                **             GETD,ChkEOT,DdlWrt,D1=SCR,F->SCR,PUTDIR,
                **             CSLC4,CSLC5,CSRC5,ASLC4,ASRC4,YMDHMS,<ENDTAP>
                **
                ** Uses.......
                **  Exclusive: A,B,C,D,R0,   R2,D1,P
                **  Inclusive: A,B,C,D,R0,R1,R2,D1,P,SCRTCH[63:0],ST[8:0]
                **
                ** Stk lvls:   4 (CLEARR)
                **
                ** History:
                **
                **    Date     Programmer          Modification
                **  --------   ----------   --------------------------------
                **  11/19/82      NZ        Added documentation
                **
                ********************************************************************
                ********************************************************************
F4326 796F  =FORMAT GOSUB  TSTAT          Check drive status
F432A 561           GONC   FORM10         OK...continue
F432D 880           ?P#    =eTAPE         Is it a drive error message?
F4330 00            RTNYES                No...must be for real
F4332 80F0          CPEX   0              Yes...check further
F4336 890           ?P=    =eNEWTA        Is it "New Medium" error?
F4339 DE            GOYES  FORMAT         Yes...try again
F433B 80F0          CPEX   0              No...
F433F 02            RTNSC                 ...Error!
                *-
                *-
F4341       FORM10                        Check if # entries is OK...
                *
                * Get # entries from R0[15:12]
                *
F4341 118           C=R0
F4344 D2            C=0    A              Clear low nibbles for rotate...
F4346 7157          GOSUB  Cslc4          ...Now C[A] is # of entries
                *
                * Convert to records and store in B[A]
                *
F434A 822           SB=0
F434D C6            C=C+C  A
F434F F6            CSR    A              Divide by 8
F4351 832           ?SB=0                 Was there a remainder?
F4354 40            GOYES  FORM20         No...continue
```

```
327 F4356 E6          C=C+1  A          Yes...increment to next record
328 F4358 AF5  FORM20 B=C    W          Copy to B[A], clear B[7:5]
329           *
330           * Get drive's maximum address (if it responds to MaxRec)
331           *
332           * Send DDT(MaxRec), ask for 2 bytes...(still talker from TSTAT)
333           *
334 F435B 20           P=     =MaxRec    Send max addressable record
335 F435D 71F6        GOSUB  Ddt        (send it)
336           *
337           * Following line removed 10/20/83 to get 3 nibbles to fix the bug
338           * noted about 15 lines below this (this RTNC is not really needed
339           * as the only two reasons that DDT will error out are 1) ATTN pre
340           * twice, and 2) Diamond has error bit set. Neither of these can
341           * change before the PUTE immediately following, so PUTE will abor
342           * with the same error.
343           *
344           *       RTNC
345           *
346 F4361 3500        LC(6)  (=mSDA)+2  Send 2 bytes!
       0000
347 F4369 8E00        GOSUBL =PUTE
       00
348 F436F 400         RTNC
349 F4372 78C4        GOSUB  Getd       Get the data byte
350 F4376 551         GONC   FORM30     OK...in C[B]
351 F4379 7C61        GOSUB  ChkEOT     Check if EOT
352 F437D 400         RTNC              If not EOT, then unexpected frame
353           *
354           * EOT...must be HP82161A (at least for size)
355           *
356 F4380 20          P=     0
357 F4382 34FF        LC(5)  511        Max record address for HP82161A
       100
358 F4389 571         GONC   FORM50     Go always
359           *-
360           *-
361           *
362           * This is a device which does respond to MaxRec...read second
363           * byte after saving first byte in A[3:2]
364           *
365 F438C DA   FORM30 A=C    A
366           *
367           * Following line is a bug fix...fixes bug with INITIALIZE for an
368           * extended Acc ID=16 protocol device and directory size
369           *
370 F438E AA0         A=0    XS         Clear the (soon-to-be) nibble 4
371 F4391 F0          ASL    A
372 F4393 F0          ASL    A
373 F4395 75A4        GOSUB  Getd       Read second data byte
374 F4399 400         RTNC
375           *
376           * Now combine the two bytes in A[3:0]
377           *
378 F439C AEA         A=C    B
```

```
    379 F439F D6          C=A    A
    380 F43A1 8AD  FORM50  ?B#0   A              Check if given dir length=0
    381 F43A4 B0          GOYES  FORM60          Not zero...leave it as is
    382             *
    383             * Specified directory length is zero...need to use default
    384             *
    385             * Default is 1/32 of total records (ignore low bits)
    386             *
    387 F43A6 D5          B=C    A              Copy total to B[A]...
    388 F43A8 E5          B=B+1  A              ...add one for zero basing...
    389 F43AA F5          BSR    A              ...divide by 16...
    390 F43AC 81D         BSRB                  ...and 2 (total 32)!
    391 F43AF       FORM60
    392             *
    393             * Now B[A] is directory length in records, R0[15:12] is length
    394             * in entries, C[A] is max addressable record address
    395             *
    396             * Check if room by the formula T - 2 - R >= N,
    397             * where T=total # of addressable records on medium (C[A]-1),
    398             *       R=# records needed for N directory entries (B[A]),
    399             *   and N=# of directory entries (R0[15:12]).
    400             *
    401 F43AF CE            C=C-1  A              Offset to total recs - 2
    402 F43B1 E9          C=C-B  A              Subtract # records needed
    403 F43B3 421         GOC    FORM65         Error!!!
    404 F43B6 110         A=R0                  Check if it passes test...
    405 F43B9 D0          A=0    A              ...Preclear high nibbles...
    406 F43BB 8E00        GOSUBL =ASLC4         ...Rotate # entries into A[A]...
            00
    407 F43C1 8BA         ?C>=A  A              ...and check for fit!
    408 F43C4 60          GOYES  FORM70         OK...continue!
    409             *
    410             * Error...out of range!
    411             *
    412 F43C6 20    FORM65 P=     =eRANGE        not OK...range error!
    413 F43C8 02          RTNSC
    414             *-
    415             *-
    416 F43CA       FORM70
    417             *
    418             * Now write the actual # of records for the directory from B[3:0]
    419             *
    420 F43CA 110         A=R0
    421 F43CD 8E00        GOSUBL =ASLC4
            00
    422 F43D3 23          P=     3
    423 F43D5 A94         A=B    WP
    424 F43D8 78C6        GOSUB  Asrc4
    425 F43DC 100         R0=A                  R0[15:12] is # of records, rest is
    426             *                           volume label
    427 F43DF 7A96        GOSUB  Mtyl
    428 F43E3 400         RTNC
    429 F43E6 20          P=     =Format
    430 F43E8 7B46        GOSUB  Ddl            Format all records of the medium
    431 F43EC 400         RTNC
```

```
 432 F43EF 70AE         GOSUB  TSTAT         Wait until finished, check status
 433 F43F3 400          RTNC                 Error formatting medium
 434            ************************************************
 435            *
 436            * Now actually write the structure on the medium...
 437            *
 438            * R0[11:0] is volume label, R0[15:12] is size of
 439            * directory in records
 440            *
 441            ************************************************
 442 F43F6 D0   =INITIL A=0    A
 443 F43F8 7BCE         GOSUB  SEEKA         Seek to first record
 444 F43FC 400          RTNC
 445 F43FF 7A76         GOSUB  Mtyl          I am going to send data
 446 F4403 400          RTNC
 447 F4406 7B26         GOSUB  DdlWrt        Set the drive to write mode
 448 F440A 400          RTNC
 449 F440D 20           P=     0
 450 F440F 3108         LCHEX  80            Disc ID (LIF standard)
 451 F4413 22           P=     2
 452 F4415 7E56         GOSUB  Putdx         ID is two bytes long
 453 F4419 400          RTNC
 454            *
 455            * Now output volume name (currently in R0[11:0])
 456            *
 457 F441C AF1          B=0    W
 458 F441F 118          C=R0
 459            *
 460            * Following 4 lines added 10/20/83 to gain 10 nibbles to fix
 461            * a bug (DDT6 bug, below) by replacing the 5 lines commented
 462            * out 10 lines down from here
 463            *
 464 F4422 8AE          ?C#0   A             Is the name zeroes?
 465 F4425 80           GOYES  INIT05        No...continue
 466 F4427 8E00         GOSUBL =BLANKC       Yes...use blanks
           00
 467 F442D      INIT05
 468            *
 469 F442D 2B           P=     11
 470 F442F A95          B=C    WP            B[11:0] is now volume label
 471 F4432 AF9          C=B    W
 472            *
 473            *       ?C#0   WP            Is the name zeroes?
 474            *       GOYES  INIT05        No...continue
 475            *       P=     0
 476            *       LCASC  \      \      Yes...set to blanks!
 477            *INIT05
 478            *
 479 F4435 8E00         GOSUBL =PRMSGA       Send the name (6 bytes)
           00
 480 F443B 400          RTNC
 481            *
 482            * Directory start address
 483            *
 484 F443E D2           C=0    A             Clear C[B]
```

```
485 F4440 23          P=    3
486 F4442 7136        GOSUB Putdx        Put first 3 bytes of dir start
487 F4446 400         RTNC
488 F4449 3120        LC(2) 2            Fourth byte of dir start is 2
489 F444D 7026        GOSUB Putd         (Start of directory is record 2)
490 F4451 400         RTNC
491           *
492           * Next four bytes required for compatibility (with 3000!!!)
493           * by the LIF standard
494           *
495 F4454 3101        LCHEX 10
496 F4458 26          P=    6
497           *
498           * Also output first two bytes of length of directory (zeros)
499           *
500 F445A 7916        GOSUB Putdx
501 F445E 400         RTNC
502           *
503           * Now get the non-zero part of directory length
504           *
505 F4461 118         C=R0
506 F4464 7336        GOSUB Cslc4        C[A] is number of records needed
507           *
508           * Output the last two bytes or directory length
509           *
510 F4468 DA          A=C   A            Save low byte in A[B]
511 F446A F6          CSR   A
512 F446C F6          CSR   A            High byte first
513 F446E 7FF5        GOSUB Putd         Send high byte
514 F4472 D6          C=A   A
515           *
516           * Output the last byte of directory length,
517           * two bytes for version number, and two
518           * required zero bytes
519           *
520           ****************************************************************
521           *                                                              *
522           * Now set version number and version 1 information...          *
523           * (Version 1 info: words 12-17, physical attributes;           *
524           * words 18-20, volume time stamp)                              *
525           *                                                              *
526           * Physical attributes:                                         *
527           * Word:            10   11   12   13   14   15   16   17     *
528           * For tape, write: 0001 0000 0000 0002 0000 0001 0000 0100     *
529           *                                                              *
530           * Volume time stamp:                                           *
531           * Word:                     18   19   20                       *
532           * For all mass mem, write:  YYMM DDHH MMSS                     *
533           *                                                              *
534           ****************************************************************
535           *
536 F4474 22          P=    2
537 F4476 7DF5        GOSUB Putdx        Output last byte of dir length
538           *                          and high byte of version number
539 F447A 400         RTNC
```

```
    540 F447D 301          LCHEX  1            (This is LIF version 1)
    541 F4480 23           P=     3
    542 F4482 71F5         GOSUB  Putdx        Output version num + zero word
    543 F4486 400          RTNC
    544             *
    545             * Determine if drive talks DDT6 here, and use that value for
    546             * device information
    547             *
    548 F4489 7F95         GOSUB  D1=SCR       Set D1 @ SCRTCH for area to write
    549 F448D 73B3         GOSUB  Ytml
    550 F4491 400          RTNC
    551             *
    552             * Following 3 lines added 10/20/83 to fix a bug with extended-
    553             * Acc ID=16 protocol devices (DDT was forgotten); adds 9 nibbles
    554             * here (pack above saves 10 nibbles...1 filler nibble added at
    555             * ChkEOT, below)
    556             *
    557 F4494 20           P=     =ImpByt      Send implementation bytes
    558 F4496 78B5         GOSUB  Ddt
    559 F449A 400          RTNC
    560             *
    561 F449D 3500         LC(6)  (=mSDA)+12   Read 12 bytes...
              0000
    562 F44A5 AFA          A=C    W
    563 F44A8 8E00         GOSUBL =PUTE        ...send message to drive
              00
    564 F44AE 400          RTNC
    565 F44B1 7983         GOSUB  Getd
    566 F44B5 534          GONC   INIT10       No carry = device did send value
    567             *
    568             * Error from GETD means either EOT or ????
    569             *
    570 F44B8 7D20         GOSUB  ChkEOT       Check if EOT
    571 F44BC 400          RTNC                No...unexpected
    572             *
    573             * Fill in the correct default values for HP82161A
    574             *
    575 F44BF AF2          C=0    W            Clear area first
    576 F44C2 1557         DAT1=C W            Clear first 16 nibbles...
    577 F44C6 17F          D1=D1+ 16
    578 F44C9 15D7         DAT1=C 8            ...and last 8...
    579 F44CD E6           C=C+1  A            ...set C[0]=1...
    580 F44CF 173          D1=D1+ 4
    581 F44D2 15D0         DAT1=C 1            Write # records per track
    582 F44D6 1C5          D1=D1- 6            Position to # surfaces/medium
    583 F44D9 15D0         DAT1=C 1            Write it
    584 F44DD 1C7          D1=D1- 8            Position to # tracks/surface
    585 F44E0 E6           C=C+1  A            Set C[0]=2
    586 F44E2 15D0         DAT1=C 1            Write it!
    587 F44E6 5B1          GONC   INIT20       Go always
    588             *-
    589             *-
    590 F44E9 80FF ChkEOT  CPEX   15           Now P is FRAME value
    591 F44ED 880          ?P#    =pEOT        Did I get an EOT?
    592 F44F0 20           GOYES  ChkEOt
```

```
593 F44F2 80FF ChkEOt  CPEX   15
594 F44F6 01           RTN
595            *-
596            *-
597 F44F8 0            CON(1) =FIXSPC       1 nibble available here
598            *-
599            *-
600            *
601            * Device did respond...C[B] is data byte (READI3 writes it
602            * at D1, increments D1 by 2, then jumps to READIT)
603            *
604 F44F9 8E00 INIT10  GOSUBL =READI3       ...into =SCRTCH (enter READIT)
          00
605 F44FF 400          RTNC                 Error
606            *
607            * Device volume information is now in SCRTCH (12 bytes)
608            *
609 F4502 7625 INIT20  GOSUB  D1=SCR        Reset D1 to =SCRTCH...
610            *
611            * First set me back as talker
612            *
613 F4506 7375         GOSUB  Mtyl
614 F450A 400          RTNC
615            *
616            * Write volume information from =SCRTCH (12 bytes)
617            *
618 F450D D2           C=0    A
619 F450F 30C          LC(1)  12
620 F4512 DA           A=C    A             Count in A[A]
621 F4514 7645         GOSUB  Writit        Send the data!
622 F4518 400          RTNC
623            *
624            * Save D0, D[A] in R2 (YMDHMS uses A-D,D0,D1,R0,R1,ST[7:0])
625            *
626 F451B 136          CDOEX
627 F451E 7675         GOSUB  Cslc5
628 F4522 DB           C=D    A
629 F4524 10A          R2=C
630            *
631            * Get creation date (current time)
632            *
633 F4527 7F75         GOSUB  Ymdhms        C[11:0] is value
634            *
635            * Save time and date in R2, restore D0, D[A]
636            *
637 F452B 12A          CR2EX
638 F452E D7           D=C    A
639 F4530 7555         GOSUB  Csrc5
640 F4534 134          D0=C
641            *
642            * Recover the time from R2 and continue
643            *
644 F4537 112          A=R2
645 F453A 8E00         GOSUBL =ASLC4        A[15:4] is value now
          00
```

```
646 F4540 26          P=     6            Send 6 characters!
647 F4542 8E00        GOSUBL =PUTALR      Send from A, start with A[15:14]
          00
648 F4548 400         RTNC
649 F454B D2          C=0    A
650 F454D 316D        LCHEX  D6           Number of bytes left to clear
651 F4551 DA          A=C    A            ...into A[A] for CLLOOP
652 F4553 76CD        GOSUB  CLLOOP       Clear this many bytes
653 F4557 400         RTNC
654 F455A 7ABD        GOSUB  CLEARN       Clear record 1 (must be 0 for LIF)
655 F455E 400         RTNC
656             *
657             * Set the first directory entry to logical end of directory
658             * (B[W] is zero from CLEARN - PUTDIR will not check status)
659             *
660 F4561 7DA4        GOSUB  F->SCR       Put "FFF"s into SCRTCH
661 F4565 8E9D        GOSUBL PUTDIR       Write a directory entry from D1
          A0
662 F456B 400         RTNC
663             *
664             * Fall through into ENDTAP!!!
665             *
666             ****************************************************************
667             ****************************************************************
668             **
669             ** Name:       ENDTAP - Clean up the loop after mass mem action
670             **
671             ** Category:   PILUTL
672             **
673             ** Purpose:
674             **      Check status of a drive, rewind it, and unaddress all
675             **      talkers and listeners
676             **
677             ** Entry:
678             **      D[X] is device address
679             **      D0 points to the mailbox
680             **
681             ** Exit:
682             **      Carry clear:
683             **        P=0, all OK
684             **      Carry set:
685             **        Error...P, C[0] are error code
686             **
687             ** Calls:      TSTAT,MTYL,DDL,<UTLEND>
688             **
689             ** Uses.......
690             **  Exclusive: C[W],P,ST[3:0]
691             **  Inclusive: C[W],P,ST[3:0]
692             **
693             ** Stk lvls:   3 (TSTAT)
694             **
695             ** History:
696             **
697             **    Date     Programmer              Modification
698             **  --------   ----------    ---------------------------------
```

```
699             ** 11/19/82      NZ          Added documentation
700             **
701             ********************************************************************
702             ********************************************************************
703             *
704             * Code above falls into this code!!!
705             *
706 F456E 712D =ENDTAP GOSUB  TSTAT          Check status of drive to finish
707 F4572 400         RTNC
708 F4575 7405        GOSUB  Mtyl
709 F4579 400         RTNC
710 F457C 2F          P=     15            Set to ignore any data sent to it
711 F457E 75B4        GOSUB  Ddl
712 F4582 400         RTNC
713 F4585 20          P=     =Rewind
714 F4587 7CA4        GOSUB  Ddl           Rewind (home) the medium
715 F458B 400         RTNC
716 F458E 8C00        GOLONG =UTLEND       Clean up the loop
        00
717             ********************************************************************
718             ********************************************************************
719             **
720             ** Name:       READR# - Read a record from mass mem into RAM
721             **
722             ** Category:   PILI/O
723             **
724             ** Purpose:
725             **      Read a specific record number
726             **
727             ** Entry:
728             **      D1 points to the destination buffer
729             **      A[3:0] contains the record number
730             **      D[X] contains the drive address
731             **      D0 points to the mailbox
732             **
733             ** Exit:
734             **      Carry clear: OK (P=0)
735             **      Carry set: Error (P, C[0] are error code)
736             **
737             ** Calls:      TSTAT,SEEKA,DdtRd,DDT,READSU,<TSTATA>
738             **
739             ** Uses.......
740             **  Exclusive:      C[W],   P
741             **  Inclusive: A[W],C[W],D1,P,ST[3:0]
742             **
743             ** Stk lvls:   3 (TSTAT)
744             **
745             ** Note: This routine will always read the device status first
746             **       and ignore any device error that is reported initially
747             **
748             ** History:
749             **
750             **    Date     Programmer          Modification
751             **  --------   ----------   ---------------------------------
752             **  08/09/83     NZ         Changed final TSTAT to TSTATA
```

```
753                ** 04/29/83      NZ          Added two buffer exchanges (cost=
754                **                           9 bytes, makes media reads faster
755                **                           and more efficient)
756                ** 04/04/83      SC          Ignore initial device error
757                ** 11/19/82      NZ          Added documentation
758                **
759                ****************************************************************
760                ****************************************************************
761 F4594 7BFC =READR# GOSUB  TSTAT              Check device status (ignore carry)
762 F4598 7B2D       GOSUB  SEEKA               Seek to that record
763 F459C 400        RTNC
764 F459F 7A94       GOSUB  DdtRd               Read that record
765 F45A3 400        RTNC
766 F45A6 20         P=     =XchgT
767 F45A8 76A4       GOSUB  Ddt                 Exchange buffers 0 and 1
768 F45AC 400        RTNC
769 F45AF 20         P=     =Read1
770 F45B1 7D94       GOSUB  Ddt                 Send data from buffer 1
771 F45B5 400        RTNC
772 F45B8 3500       LC(6)  (=MSDA)+#100        #100 bytes = 1 record
           0000
773                *
774                * Read one record from the drive to the buffer (D1)
775                *
776 F45C0 77A4       GOSUB  Readsu              Read from drive to (D1)
777 F45C4 400        RTNC
778 F45C7 20         P=     =XchgT
779 F45C9 7584       GOSUB  Ddt                 Exchange buffers 0 and 1 back
780 F45CD 400        RTNC
781                *
782                * When here, all 256 bytes have been read
783                *
784 F45D0 69CC       GOTO   TSTATA              Check final device status
785                ****************************************************************
786                ****************************************************************
787                **
788                ** Name:       WRITE# - Write to a specific record
789                **
790                ** Category:   PILI/O
791                **
792                ** Purpose:
793                **      Write to a specific record on a mass mem device
794                **
795                ** Entry:
796                **      D1 points to the input buffer
797                **      A[3:0] contains the record number to be written
798                **      D[X] contains the drive address
799                **      D0 points to the mailbox
800                **
801                ** Exit:
802                **      Carry clear if OK (P=0)
803                **      Carry set if error (P, C[0] are error code)
804                **
805                ** Calls:      TSTAT,SEEKA,MTYL,DdlWrt,DDL,WRITIT
806                **
```

```
                ** Uses.......
                **  Exclusive: A[A],        P
                **  Inclusive: A[A],C[W],D1,P,ST[8],ST[3:0]
                **
                ** Stk lvls:   3 (TSTAT)
                **
                ** Note: This routine always reads the device status first and
                **       ignores any initial device error.
                **
                ** History:
                **
                **    Date     Programmer          Modification
                **  --------   ----------   --------------------------------
                **  04/04/83      SC        Ignore initial device error
                **  11/19/82      NZ        Added documentation
                **
                ****************************************************************
                ****************************************************************
F45D4 7BBC =WRITE# GOSUB  TSTAT          Check device status (ignore carry)
F45D8 7BEC         GOSUB  SEEKA
F45DC 400          RTNC
F45DF 7A94         GOSUB  Mtyl
F45E3 400          RTNC
F45E6 7B44         GOSUB  DdlWrt         Set drive to write mode
F45EA 400          RTNC
F45ED D0           A=0    A
F45EF B24          A=A+1  XS             A[A]=#00100 (1 record)
                *
                * Transfer 256 bytes (one record)
                *
F45F2 7864         GOSUB  Writit
F45F6 400          RTNC
F45F9 2F           P=     15             DDL15 = Ignore data!
F45FB 7834         GOSUB  Ddl            (Ignore data)
F45FF 400          RTNC
F4602 609C         GOTO   TSTAT          Check status, exit
                ****************************************************************
                ****************************************************************
                **
                ** Name:       MOVEFL - Move a file between two HPIL devices
                **
                ** Category:   PILI/O
                **
                ** Purpose:
                **      Move a block of "records" from one HPIL device to
                **      another
                **
                ** Entry:
                **      R1[A] = device addr of destination device (from FILSPx)
                **      R2[A] = device addr of source device (from FILSPx)
                **      R3[A] = record address of destination if mass mem
                **      B[A]  = record address of source if mass mem
                **      R3[9:5] = number of records to copy
                **
                ** Exit:
```

```
                **      P#0!
                **      Carry clear: OK
                **      Carry set: error (P, C[0] are error code)
                **
                ** Calls:      CSLC5,D1=AVE,CSRC10,CSLC10,START,GETDev,SEEKA,
                **             CHKBIT,DdtRd,READSU,D1@AVS,CSRC5,MTYL,DDL,ASRC10,
                **             WRITIT,hCPY5s,ASRC5,YTML
                **
                ** Uses.......
                **  Exclusive: A[W],C[W],D[A],R3[14:10],R4,D0,D1,P,ST[4:0]
                **  Inclusive: A[W],C[W],D[W],R3[14:10],R4,D0,D1,P,ST[8],ST[4:0]
                **
                ** Stk lvls:   3 (SEEKA)(hCPY5s)
                **
                ** Detail:
                **      COUNT# is R3[14:10]  - # of records this transfer
                **      COUNTD is R4[9:5]    - # of records already finished
                **      COUNTR is R4[14:10]  - # of records remaining
                **      COUNT  is R3[9:5]    - # of records to move (total)
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **  --------   ----------   ---------------------------------
                **  08/29/83      NZ        Changed where I set up A[A] for
                **                          the source so that the call to
                **                          START doesn't destroy # records
                **  08/19/83      NZ        Added checks for device mode and
                **                          changed calls to FNDMB+ to START
                **  05/25/83      NZ        Added checks for mass mem...if not
                **                          mass mem, then just move bytes
                **  01/14/83      NZ        Fixed several bugs!
                **  01/10/83      NZ        Added documentation
                **
                ********************************************************************
                ********************************************************************
F4606           =MOVEFL
F4606 11B               C=R3
F4609 D2                C=0    A
F460B 7984              GOSUB  Cslc5          Save # of records in R4[14:10]
F460F 10C               R4=C                  Save record count in R4[9:5]!
                *
                * R4[9:5] is the count of how many records I have moved,
                * R4[14:10] is # of records remaining
                *
F4612 8E00 MOVEF1       GOSUBL =D1=AVE        Set D1=AVMEME
      00
F4618 147               C=DAT1 A
F461B 1C4               D1=D1- 5              Point to AVMEMS
F461E AF0               A=0    W              Clear high nibs for ASRB
F4621 143               A=DAT1 A
F4624 131               D1=A                  Set D1 @ AVMEMS
                *
                * AVMEME in C[A], AVMEMS in A[A]
                *
```

```
916 F4627 E2          C=C-A  A          C[A] is # nibbles available
917 F4629 DA          A=C    A
918 F462B 81C         ASRB             A[A] is # bytes available
919 F462E F4          ASR    A
920 F4630 F4          ASR    A          A[A] is # records available
921 F4632 20          P=     =eNORAM
922 F4634 8A8         ?A=0   A
923 F4637 00          RTNYES            Error...memory too small
924             *
925             * A[A] is # of records to copy at a chunk, D1 @ AVMEMS
926             *
927 F4639 11C         C=R4
928 F463C 7554        GOSUB  Csrc10     Now C[A] is # of records left
929 F4640 8AE         ?C#0   A
930 F4643 40          GOYES  MOVEF2     Not done...continue
931 F4645 03          RTNCC             Done...return, carry clear
932             *-
933             *-
934 F4647 E2  MOVEF2  C=C-A  A
935 F4649 560         GONC   MOVEF3     If no carry, not done
936 F464C CA          A=A+C  A          Set A=old C (A+(C-A) = C)
937 F464E D2          C=0    A          Set remaining count = 0
938 F4650 7444 MOVEF3 GOSUB  Cslc5
939             *
940             * Pause here to set COUNT# (R3[14:10]) to COUNTA(A[A])
941             *
942 F4654 12B         CR3EX
943 F4657 7A34        GOSUB  Csrc10
944 F465B D6          C=A    A          Copy COUNTA to COUNT#
945 F465D 8E00        GOSUBL =CSLC10
          00
946 F4663 12B         CR3EX             Restore C, R3 (with new value)
947             *
948             * Now continue on...(C[A] is number of records done)
949             *
950 F4666 7E24        GOSUB  Cslc5
951 F466A 10C         R4=C              Write the counts back out
952             *
953             * Copy the nibbles...need to call SETUP every time...
954             *       increment position by # records moved
955             *
956 F466D 11A         C=R2              Get source address
957 F4670 D7          D=C    A
958 F4672 7D04        GOSUB  Start      Set up for src, find that mailbox
959 F4676 400         RTNC              Not found...error!
960             *
961             * Set A[A] to the number of records done
962             *
963 F4679 114         A=R4
964 F467C 8E00        GOSUBL =ASRC5     A[A]=# records done
          00
965             *
966             * First check if in device mode (if so, just send data)
967             *
968 F4682 8E00        GOSUBL =GETDev    Check if device mode
```

```
           00
  969 F4688 412          GOC    MOVEd1        Device mode...just send data
  970              *
  971              * Check if this is a mass mem or other device
  972              *
  973 F468B 7F7C         GOSUB  CHKBIT        If mass mem, carry set
  974 F468F 4A0          GOC    MOVEF,        Mass mem...Seek, Read
  975 F4692 7EA1         GOSUB  Ytml          Not mass mem...just make me talker
  976 F4696 6010         GOTO   MOVEF4        Check carry, continue
  977              *-
  978              *-
  979              *
  980              * A[A] is # records offset to file data
  981              *
  982 F469A C0   MOVEF,  A=A+B  A             Get source record #
  983 F469C 7AA3         GOSUB  Seeka         Go to that record
  984 F46A0 400          RTNC
  985 F46A3 7693         GOSUB  DdtRd         Read the data from the drive
  986 F46A7 400  MOVEF4  RTNC
  987 F46AA 11B  MOVEd1  C=R3                 Now get COUNT# back from R3[14:10]
  988 F46AD 74E3         GOSUB  Csrc10
  989 F46B1 F2           CSL    A
  990 F46B3 F2           CSL    A             Convert COUNT# to BYTES
  991 F46B5 8E00         GOSUBL =hCPY5s       Set up for SDA/SFC message
           00
  992 F46BB 7CA3         GOSUB  Readsu        Read after set-up
  993 F46BF 400          RTNC                 Error!
  994              *
  995              * Now have the data in RAM, starting at AVMEMS!
  996              *
  997 F46C2 8E00         GOSUBL =D1@AVS       Set D1 to (AVMEMS)
           00
  998 F46C8 119          C=R1                 Get the destination address
  999 F46CB D7           D=C    A
 1000 F46CD 72B3         GOSUB  Start         Find the destination mailbox
 1001 F46D1 400          RTNC
 1002 F46D4 8E00         GOSUBL =GETDev       Check if device mode
           00
 1003 F46DA 413          GOC    MOVEF6        Yes...just send data
 1004 F46DD 7D2C         GOSUB  CHKBIT        Check if mass storage
 1005 F46E1 551          GONC   MOVEF5        Not mass storage...skip Seek
 1006 F46E4 11C          C=R4
 1007 F46E7 7E93         GOSUB  Csrc5         Now COUNTD is in C[A]
 1008 F46EB 113          A=R3                 A[A] is dest address
 1009             *
 1010             * Now C[A] is COUNTD (done), A[A] is dest address
 1011             *
 1012 F46EE CA           A=A+C  A             A[A] is desired address!
 1013 F46F0 7653         GOSUB  Seeka         Seek to that record
 1014 F46F4 400          RTNC
 1015 F46F7 7283 MOVEF5  GOSUB  Mtyl          I am talker now
 1016 F46FB 400          RTNC
 1017 F46FE 7C0C         GOSUB  CHKBIT        Check again if mass storage
 1018 F4702 590          GONC   MOVEF6        Not mass storage...skip Write
 1019 F4705 7C23         GOSUB  DdlWrt
```

```
1020 F4709 400          RTNC
1021 F470C 113  MOVEF6  A=R3
1022 F470F 8E00         GOSUBL =ASRC10       Get COUNT# from R3[14:10]
           00
1023               *
1024               * A[A] is now the count in records, D1 @ AVMEMS
1025               *
1026 F4715 11C          C=R4
1027 F4718 7D63         GOSUB  Csrc5         C[A] is now COUNTD (done)
1028 F471C C2           C=C+A  A             Update COUNTD to new value
1029 F471E 7673         GOSUB  Cslc5
1030 F4722 10C          R4=C                 Write it back out!
1031 F4725 F0           ASL    A
1032 F4727 F0           ASL    A             A[A] is # bytes now
1033 F4729 7133         GOSUB  Writit        Send the data to the drive!
1034 F472D 400          RTNC
1035 F4730 61EE         GOTO   MOVEF1        Loop back to finish if more
1036               ********************************************************************
1037               ********************************************************************
1038               **
1039               ** Name:       FINDFL - Set up loop, get a directory entry
1040               ** Name:       FINDF+ - Set up loop, get directory entry (MS)
1041               ** Name:       FINDFx - Find a file on a mass storage device
1042               **
1043               ** Category:   FILUTL
1044               **
1045               ** Purpose:
1046               **      Find file on external device (for FINDF+ and FINDFx,
1047               **      the device must be a mass storage device)
1048               **
1049               ** Entry:
1050               **      FINDFL,FINDF+:
1051               **        First 8 characters in A[W], last 2 in R0[3:0]
1052               **        D[A] is device address (set up by FILSPx poll handler)
1053               **      FINDFx:
1054               **        D[X] is mass storage device address
1055               **        D0 points to the mailbox
1056               **        First 8 chars of name in R0, last 2 in R1[3:0]
1057               **
1058               ** Exit:
1059               **      Carry clear:
1060               **        File directory entry in =SCRTCH[32]
1061               **        A[A] is starting record (A[4]=0)
1062               **        C[A] is number of records (C[4]=0)
1063               **        D1 points to file type
1064               **        B[3:0] is directory pointer for file (B[3:1] is
1065               **          record number, B[0] is entry within record)
1066               **      Carry set:
1067               **        P=0: Names don't match (same conditions as carry clear)
1068               **        P#0: Error (P, C[0] are error code)
1069               **
1070               ** Calls:      START,CHKBIT,CHKMAe,YTML,D1=SCR,READSU,hCPY5s,
1071               **  FINDFx --> GETDR!,NXTEN+,CSRC5,CSLC5,GETDIR,GETZER
1072               **
1073               ** Uses.......
```

```
1074            ** Exclusive: A,B,C,        D1,P,             ST[5]
1075            ** Inclusive: A,B,C,D[15:5],D1,P,SCRTCH[63:0],ST[5:0]
1076            **
1077            ** Stk lvls:   5 (GETDR!)
1078            **
1079            ** History:
1080            **
1081            **    Date     Programmer            Modification
1082            **  --------   ----------   --------------------------------
1083            **  10/07/83      NZ        Updated documentation
1084            **  05/25/83      NZ        Added check for mass storage, not
1085            **                          Acc ID=16 (if true, RTNSXM)
1086            **  05/12/83      NZ        Removed call to CHKMAS, replaced
1087            **                          with call to CHKBIT (checks bits
1088            **                          from FILSPx); removed CONWUC call
1089            **  02/11/83      NZ        Added ST(Loop?)
1090            **  11/19/82      NZ        Added documentation
1091            **
1092            ****************************************************************
1093            ****************************************************************
1094 F4734 850  =FINDFL ST=1   =sLoop?       LOOP is allowed for FINDFL
1095 F4737 6600         GOTO   FINDf+
1096            *-
1097            *-
1098 F473B 840  =FINDF+ ST=0   =sLoop?       LOOP not allowed for FINDF+
1099 F473E      FINDf+
1100 F473E 120          AROEX                Save first 8 chars in R0
1101 F4741 101          R1=A                 Save last 2 chars in R1
1102 F4744 7B33         GOSUB  Start         Set up the transfer!
1103 F4748 400          RTNC                 Error...return!
1104 F474B 96B          ?D=0   B             Is this "LOOP"?
1105 F474E 56           GOYES  FINDF1        Yes...just read 32 bytes, check
1106            *
1107 F4750 7ABB         GOSUB  CHKBIT        Check if Acc ID=16 bit set
1108 F4754 427          GOC    FINDFx        Mass storage...continue
1109            *
1110            * If here, need to check sLoop?...if NOT set, then error!
1111            *
1112 F4757 7AAB         GOSUB  CHKMAe        Set up device type error...
1113 F475B 860          ?ST=0  =sLoop?       ...check if needed!
1114 F475E 00           RTNYES               Error!!! (Set up by CHKMAe)
1115            *
1116            * Device is OK here...just read in the directory info!
1117            *
1118 F4760 70E0         GOSUB  Ytm1          Device is talker
1119 F4764 400          RTNC
1120 F4767 3500         LC(6)  (=mSDA)+32    Directory length is 32 bytes
           0000
1121 F476F 79B2 FIND12  GOSUB  D1=SCR
1122 F4773 74F2         GOSUB  Readsu        Save length in A[A], read data
1123 F4777 400          RTNC                 Error if carry!
1124            *
1125            * Now check if the name is OK or not...
1126            *
1127 F477A 110          A=R0                 Recall first 8 chars
```

```
1128 F477D 1D00          D1=(2) =SCRTCH       Move to name field
1129 F4781 1577          C=DAT1 W             Pre-read name
1130 F4785 17F           D1=D1+ 16            Move to 9th and 10th char of name
1131 F4788 D1            B=0    A             Clear directory pointer first!
1132 F478A 8A8           ?A=0   A             Name specified?
1133 F478D 51            GOYES  FIND14        No...accept it regardless of value
1134 F478F 976           ?A#C   W             Different name?
1135 F4792 71            GOYES  FINDfn        Yes...error (Names don't match)
1136 F4794 111           A=R1                 No...check last 2 chars
1137 F4797 D6            C=A    A             (Copy C[4])
1138 F4799 15F3          C=DAT1 4             Read last 2 chars
1139 F479D 8A6           ?A#C   A             Last 2 chars match?
1140 F47A0 90            GOYES  FINDfn        No...error (Names don't match)
1141 F47A2 173  FIND14   D1=D1+ 4             Yes...position to TYPE
1142 F47A5 67A0          GOTO   FINDF4        Set up exit conditions and exit
1143             *-
1144             *-
1145 F47A9 75FF FINDfn   GOSUB  FIND14        Set up A,C (P=0 before call)
1146 F47AD 02            RTNSC                P#0 if too big, else bad name
1147             *-
1148             *-
1149 F47AF 20   FINDle   P=     =eDSPEC       Device spec error (LOOP)
1150 F47B1 02            RTNSC
1151             *-
1152             *-
1153 F47B3 860  FINDF1   ?ST=0  =sLoop?       Is LOOP allowed?
1154 F47B6 9F            GOYES  FINDle        No...error!
1155 F47B8 D2            C=0    A
1156 F47BA 3102          LC(2)  32            Read 32 bytes from Diamond
1157 F47BE 8E00          GOSUBL =hCPY5s       Set for frame count/SDA
          00
1158 F47C4 5AA           GONC   FIND12        Go always
1159             *-
1160             *-
1161             *
1162             * Find the file on the mass storage device
1163             *
1164 F47C7 840  =FINDFx ST=0   =sLoop?       If here, this cannot be LOOP!
1165 F47CA 7E90          GOSUB  GETDR!        Get directory start, first entry
1166 F47CE 400           RTNC                 Error
1167             *
1168             * Entry name in A[W], D1 points to last 2 chars
1169             *
1170 F47D1 173  FINDF0   D1=D1+ 4             Skip last 2 chars
1171             *
1172             * Both the EOD mark (#FFFF) and PURGED file type (#0000) are
1173             * symmetric bytewise, so I can speed up the search and save
1174             * code by just reading the value straight from RAM (not swapping
1175             * the bytes as I normally should)
1176             *
1177 F47D4 15F3          C=DAT1 4             Read in the type
1178 F47D8 23            P=     3
1179 F47DA B16           C=C+1  WP            Check for end of directory
1180 F47DD 415           GOC    FINDFn        File not found!
1181 F47E0 A1E           C=C-1  WP            Check for purged file
```

```
1182 F47E3 91A          ?C=0   WP
1183 F47E6 F1           GOYES  FINDF1        PURGED!
1184             *
1185             * Now check if names match
1186             *
1187 F47E8 118          C=RO
1188 F47EB 976          ?A#C   W             Check first 8 chars
1189 F47EE 71           GOYES  FINDF1
1190 F47F0 1C3          D1=D1- 4
1191 F47F3 15F3         C=DAT1 4
1192 F47F7 173          D1=D1+ 4             Leave D1 @ type!
1193 F47FA 121          AR1EX                Now check last 2 chars
1194 F47FD 912          ?A=C   WP
1195 F4800 A4           GOYES  FINDF3        MATCH!
1196 F4802 121          AR1EX                Get back directory information
1197 F4805       FINDF1
1198             *
1199             * This is NOT the file! Get directory ptr from B[3:0]...
1200             *
1201 F4805 78A2         GOSUB  NXTEN+        Get next entry (carry if new rec)
1202 F4809 D5           B=C    A             Store back in B[3:0]
1203 F480B 5A1          GONC   FINDF2        Not new record...read next entry
1204             *
1205             * Next record needed...check if reached physical EOD yet
1206             *
1207 F480E AFB          C=D    W
1208 F4811 7472         GOSUB  Csrc5         Directory length in C[3:0]
1209 F4815 23           P=     3
1210 F4817 A1E          C=C-1  WP            Decrement record count...
1211 F481A 91A          ?C=0   WP            More records?
1212 F481D 21           GOYES  FINDFn        No...file not found (EOD)
1213 F481F 7572         GOSUB  Cslc5         Yes...read next record
1214 F4823 AF7          D=C    W             Save count back in D[8:5]
1215             *
1216             * Now read next entry, loop back
1217             *
1218 F4826 7B80 FINDF2  GOSUB  GETDIR        Read next entry after status!
1219 F482A 56A          GONC   FINDFO        (Can pack this by GOTO, move
1220             *                            FINDFO up one line)
1221 F482D 02           RTNSC                Error!
1222             *-
1223             *-
1224 F482F       FINDFn
1225             *
1226             * File not found
1227             *
1228 F482F 20           P=     0
1229 F4831 300          LC(1)  =eNFILE       File not found...
1230 F4834 20           P=     =eTAPE        ...drive error!
1231 F4836 02           RTNSC
1232             *-
1233             *-
1234 F4838 8C00 Getzer  GOLONG =GETZER       Read 4 bytes, check first two=0
          00
1235             *-
```

```
1236                *-
1237 F483E 8C00 Getd    GOLONG =GETD
          00
1238                *-
1239                *-
1240 F4844 8C00 Ytml    GOLONG =YTML
          00
1241                *-
1242                *-
1243 F484A 121  FINDF3  AR1EX               Save last 2 chars of name again
1244                *
1245                * Found the file (D1 is at file type)
1246                *
1247 F484D 173  FINDF4  D1=D1+ 4             Skip to start address field
1248 F4850 74EF         GOSUB  Getzer       Read 4 bytes, check first two=0
1249 F4854 431          GOC    FINDFe       Error (First two bytes # 0)
1250 F4857 DA           A=C    A            Save start address in A[3:0]
1251                *
1252                * Now get the length in records
1253                *
1254 F4859 7BDF         GOSUB  Getzer       Read 4 bytes, check first two=0
1255 F485D 4A0          GOC    FINDFe       Error (First two bytes # 0)
1256 F4860 1CF          D1=D1- 16           Move back to start address...
1257 F4863 1C3          D1=D1- 4            ...and back to file type
1258 F4866 03           RTNCC               Done!
1259                *-
1260                *-
1261 F4868      FINDFe
1262                *
1263                * Argument out of range
1264                *
1265 F4868 20           P=     =eRANGE
1266 F486A 02           RTNSC
1267                ****************************************************************
1268                ****************************************************************
1269                **
1270                ** Name:       GETDR! - Get first directory entry from drive
1271                ** Name:       GETDIR - Get the next directory entry from drive
1272                ** Name:       GETDR" - Get the next directory entry @ B[3:0]
1273                ** Name:       GETDR# - Get the next directory entry @ A[3:0]
1274                ** Name:       GETDR+ - Get the next directory entry @ A[S]
1275                **
1276                ** Category:   FILUTL
1277                **
1278                ** Purpose:
1279                **      GETDR!: Get the first entry in an LIF directory
1280                **      GETDR": Get the B[3:0]th entry in an LIF directory
1281                **      GETDR#: Get the A[3:0]th entry in an LIF directory
1282                **      GETDR+: Get the A[S] entry in the current record
1283                **      GETDIR: Get the next entry in an LIF directory
1284                **
1285                ** Entry:
1286                **      D[X] is the drive address
1287                **      D0 points to the mailbox
1288                **      GETDIR: Drive is addressed as talker, me as listener
```

```
1289                **      GETDR": B[3:0] is the directory entry #
1290                **      GETDR#: A[3:0] is the directory entry #
1291                **      GETDR+: A[S] is the directory offset nibble in record
1292                **
1293                ** Exit:
1294                **      Carry clear:
1295                **        Directory entry in =SCRTCH[32]
1296                **        A[W] is first 8 chars of filename
1297                **        D1 points past first 8 chars of filename
1298                **      Carry set:
1299                **        Error (P, C[0] are error code)
1300                **
1301                ** Calls:      GDIRST,SEEKA,DDT,MTYL,PUTD,YTML,TSTATA,READSC,
1302                **              D1=SCR
1303                **
1304                ** Uses.......
1305                **  Exclusive: A,  C,        P
1306                **  Inclusive: A,B,C,D[15:5],P,SCRTCH[63:0],ST[4:0]
1307                **
1308                ** Stk lvls:   GETDR!: 4 (GDIRST)
1309                ** Stk lvls:   GETDR": 3 (SEEKA)(TSTATA)
1310                ** Stk lvls:   GETDR#: 3 (SEEKA)(TSTATA)
1311                ** Stk lvls:   GETDR+: 3 (TSTATA)
1312                ** Stk lvls:   GETDIR: 3 (TSTATA)
1313                **
1314                ** History:
1315                **
1316                **    Date     Programmer              Modification
1317                **  --------   ----------    ---------------------------------
1318                **  11/19/82      NZ         Added documentation
1319                **
1320                ********************************************************************
1321                ********************************************************************
1322 F486C 7860 =GETDR! GOSUB  GDIRST        Get directory start
1323 F4870 400         RTNC
1324 F4873 D4   =GETDR" A=B    A
1325 F4875 814  =GETDR# ASRC                 Save BP value in A[S]
1326 F4878 AD0         A=0     M             Clear high nibble for SEEK
1327 F487B 784A        GOSUB   SEEKA         Go to that record
1328 F487F 400         RTNC
1329 F4882 77B1        GOSUB   DdtRd         Read that record (Drive is talker)
1330 F4886 400         RTNC
1331 F4889 948         ?A=0    S             Is the BP to be zero?
1332 F488C 92          GOYES   GETDIR        Yes...skip setting it!
1333 F488E 7BE1 =GETDR+ GOSUB  Mtyl          I must be talker for this!
1334 F4892 400         RTNC
1335 F4895 20          P=      =SetBP
1336 F4897 7C91        GOSUB   Ddl           Set byte pointer command
1337 F489B 400         RTNC
1338 F489E 810         ASLC                  Get pointer in A[0]
1339 F48A1 D6          C=A     A             Copy A[0] to C[0]
1340 F48A3 F2          CSL     A             Entry * 16
1341 F48A5 C6          C=C+C   A             Entry * 32
1342 F48A7 76C1        GOSUB   Putd          Send the Byte pointer value
1343 F48AB 400         RTNC
```

```
1344 F48AE 729F         GOSUB  Ytml          I am listener!
1345 F48B2 400          RTNC
1346              *
1347              * Drive should already be talker for GETDIR!
1348              *
1349 F48B5 71E9 =GETDIR GOSUB  TSTATA        Check if successful read!
1350 F48B9 400          RTNC
1351 F48BC 3500         LC(6)  (=MSDA)+32    Length of one directory entry
           0000
1352 F48C4 7F91         GOSUB  Readsc        Read into scratch RAM!
1353 F48C8 400          RTNC                 Error!
1354 F48CB 7D51         GOSUB  D1=SCR        Go back to SCRTCH...
1355 F48CF 1537         A=DAT1 W             Read the first 8 chars of name...
1356 F48D3 17F          D1=D1+ 16            Skip name field...
1357 F48D6 03           RTNCC                And return!
1358              ****************************************************************
1359              ****************************************************************
1360              **
1361              ** Name:       GDIRST - Get directory start and information
1362              **
1363              ** Category:   FILUTL
1364              **
1365              ** Purpose:
1366              **      Locate the start of directory (and length) on mass mem
1367              **      and return both to the caller
1368              **
1369              ** Entry:
1370              **      D[X] contains the drive address
1371              **      DO points to the mailbox
1372              **
1373              ** Exit:
1374              **      Carry clear:
1375              **        B[W] contains:
1376              **          Directory start pointer in [3:0], [15:12]
1377              **          Start of data area in [7:4]
1378              **          Zero in [11:8]
1379              **        D[W] contains:
1380              **          Drive address in [A] (No change)
1381              **          Number of directory records in [8:5]
1382              **          Address of LAST data record + 1 [12:9]
1383              **          Zero in [15:13]
1384              **      Carry set:
1385              **        Error (P, C[0] are error code)
1386              **
1387              ** Calls:      SEEKA,DdtRd,READSC,D1=SCR,GETALR,ASLC9,ASRC4,
1388              **             GETZER,(GDIRSM),ASRC9,CSRC8,ASRC3,ASLC3,CSLC4
1389              **
1390              ** Uses.......
1391              **  Exclusive: A,B,C,D[15:5],D1,P
1392              **  Inclusive: A,B,C,D[15:5],D1,P,SCRTCH[63:0],ST[3:0]
1393              **
1394              ** Stk lvls:   3 (SEEKA)(GDIRSB)
1395              **
1396              ** History:
1397              **
```

```
1398                    **    Date     Programmer          Modification
1399                    **  --------   ----------   ------------------------------
1400                    **  11/19/82      NZ        Added documentation
1401                    **
1402                    ****************************************************************
1403                    ****************************************************************
1404 F48D8 D0     =GDIRST A=0     A
1405 F48DA 79E9        GOSUB   SEEKA           (Leaves drive as talker)
1406 F48DE 400         RTNC
1407 F48E1 7851        GOSUB   DdtRd           Read medium at current record
1408 F48E5 400         RTNC
1409 F48E8 20          P=      0
1410 F48EA 3500        LC(6)   (=mSDA)+24      Read LIF ID, label, start addr,
          0000
1411                *                           length, version #, Secondary ID
1412 F48F2 7171        GOSUB   Readsc
1413 F48F6 400         RTNC                    Error...bad read
1414 F48F9 7F21        GOSUB   D1=SCR          Reset D1 to start of data
1415 F48FD 22          P=      2
1416 F48FF 8E00        GOSUBL  =GETALR         Get LIF ID
          00
1417                *
1418                * Check if this is an LIF format medium (LIF ID=#8000)
1419                *
1420 F4905 3300        LCHEX   8000
          08
1421 F490B 23          P=      3
1422 F490D 916         ?A#C    WP
1423 F4910 F1          GOYES   GDIRSe          Not LIF...error
1424 F4912 17B  GDIRS1 D1=D1+  12              Skip volume label (ignore)
1425 F4915 AF0         A=0     W
1426 F4918 24          P=      4
1427 F491A 8E00        GOSUBL  =GETALR         Get start address of directory
          00
1428 F4920 958         ?A=0    M               If any but low 3 nibs#0, error!
1429 F4923 11          GOYES   GDIRS3          OK!
1430 F4925 20   GDIRSE  P=      =eTSIZE         Error!
1431 F4927 80F0 GDIRsE CPEX    0
1432 F492B 20          P=      =eTAPE          Drive error (Size of File)
1433 F492D 02          RTNSC
1434                *-
1435                *-
1436 F492F      GDIRSe
1437 F492F 20          P=      =eNOLIF         Not LIF!
1438 F4931 45F         GOC     GDIRsE          Go always
1439                *-
1440                *-
1441 F4934 8E00 GDIRS3 GOSUBL  =ASLC9
          00
1442                *
1443                * A=[<--000--> <--Directory start address--> <--000-->]
1444                *    15.....12,11.....................9,8.......0
1445                *
1446                *
1447                * Now read number of records in the directory
```

```
1448                *
1449 F493A 177             D1=D1+ 8            Skip unneeded info in header
1450 F493D 15B3            A=DAT1 4            Read first two bytes of length
1451 F4941 173             D1=D1+ 4            Skip past them...
1452 F4944 8AC             ?A#0   A
1453 F4947 ED              GOYES  GDIRSE         Too big!
1454 F4949 22              P=     2            Read 2 bytes...
1455 F494B 8E00            GOSUBL =GETALR      Read the last two bytes of length
           00
1456                *
1457                * A=[<--Dir start address--> <--0000--> <--Dir length-->]
1458                *    15.................13,12.......4,3.............0
1459                *
1460 F4951 7F41            GOSUB  Asrc4
1461                *
1462                * A=[<--Dir length-->,<--Dir start address-->,<--000-->]
1463                *    15............12,11...................9,8.......0
1464                *
1465                * Now get the extension field...if extension > 0, read it!
1466                *
1467 F4955 D2              C=0    A            Clear high nibble...
1468 F4957 15F3            C=DAT1 4            ...Read in the extension...
1469 F495B 8AE             ?C#0   A            ...is it zero (no extensions)?
1470 F495E A0              GOYES  GDIRS4       No...read it.
1471                *
1472                * Extension field=0...fill in the default value for tape end
1473                *
1474 F4960 3200            LC(3)  #200         First record past tape
           2
1475 F4965 5C3             GONC   GDIRS8       Go always!
1476                *-
1477                *-
1478 F4968 3500  GDIRS4    LC(6)  (=MSDA)+12   Send 12 bytes from here...
           0000
1479 F4970 73F0            GOSUB  Readsc       ...to SCRTCH!
1480 F4974 400             RTNC                Error!
1481                *
1482                * READSC uses A[5:0] only
1483                *
1484 F4977 1E00            D1=(4) (=SCRTCH)+16
           00
1485 F497D 77BE            GOSUB  Getzer
1486 F4981 491             GOC    GDIRS7       Too big...use #FFFF
1487                *
1488                * Put # of records per track into A[A]
1489                *
1490 F4984 DA              A=C    A
1491                *
1492                * A[3:0] is # of records per track, A[4]=0
1493                *
1494 F4986 1CF             D1=D1-  16          Point to surfaces/medium
1495                *
1496                * Call subroutine to get surfaces/medium and multiply times
1497                * records per track (result in A[3:0])
1498                *
```

```
1499 F4989 7060          GOSUB  GDIRSM
1500 F498D 4D0           GOC    GDIRS7          Too big...use #FFFF
1501               *
1502               * A is now (records/track) * (surfaces/medium)
1503               *
1504 F4990 7890          GOSUB  D1=SCR          Tracks/surface
1505               *
1506               * Get tracks/surface, multiply times (records/track *
1507               *                                     surfaces/medium)
1508               *
1509 F4994 7550          GOSUB  GDIRSM
1510               *
1511               * Now A[3:0] is tracks/medium! (= last rec #)
1512               *
1513               * A=[<-Dir length->,<-Dir start addr->,<-0->,X,<-last rec #->]
1514               *    15.....12,11.....9,8...5,4,3.....0
1515               *
1516 F4998 5B0           GONC   GDIRS9          All OK if no carry
1517 F499B D2    GDIRS7  C=0    A               More than I can do...use #FFFF!
1518 F499D 23            P=     3
1519 F499F A1E           C=C-1  WP              Default value! (#FFFF)
1520 F49A2 DA    GDIRS8  A=C    A               C[3:0] is # of records in dir
1521 F49A4       GDIRS9
1522 F49A4 8E00          GOSUBL =ASRC9          Roll to correct fields for return
          00
1523               *
1524               * A=[<-0->,<-last rec #->,<-dir length->,<-dir start addr->]
1525               *    15.11,10.....7,6.....3,2.....0
1526               *
1527 F49AA AF2           C=0    W
1528 F49AD AB6           C=A    X               C[X] is dir start address
1529 F49B0 F2            CSL    A               Set record pntr to zero (first)
1530 F49B2 D5            B=C    A               Set PTRC to Directory start
1531 F49B4 8E00          GOSUBL =CSRC8          Shift directory start to [11:8]
          00
1532               *
1533               * PTRF area is now in C[3:0]...
1534               *
1535 F49BA AB6           C=A    X               Copy directory start to C[3:0]
1536 F49BD 8E00          GOSUBL =ASRC3          Rotate directory length to A[3:0]
          00
1537 F49C3 23            P=     3
1538 F49C5 A12           C=C+A  WP              Now C[3:0] is PTRF initial value
1539 F49C8 8E00          GOSUBL =ASLC3          Rotate A[W] back where it belongs
          00
1540 F49CE 79C0          GOSUB  Cslc4
1541 F49D2 A99           C=B    WP              Copy PTRC (set up) to C[3:0]...
1542 F49D5 AF5           B=C    W               ...and finish setting all PTRs
1543               *
1544               * Now set PFC, Dlenl, NEW, PhEOD, and Tendr
1545               *
1546 F49D8 AF6           C=A    W               Directory length and medium end...
1547 F49DB BF2           CSL    W               ...shift...
1548 F49DE BF2           CSL    W               ...to C[8:5]...
1549 F49E1 2C            P=     12
```

```
1550 F49E3 DB          C=D    A            ...copy D[A] to C[A]...
1551 F49E5 AF3         D=0    W            ...clear high nibbles of D...
1552             *                          ...(PFC, NEW, PhEOD)...
1553 F49E8 A97         D=C    WP           ...and copy it all to D!
1554             *
1555             * Done with initialization!
1556             *
1557 F49EB 03          RTNCC
1558             *-
1559             *-
1560             *
1561             * This is the routine to get from RAM & multiply by A[3:0]
1562             * (Uses A[A], C[A], D1, P!!) (P is NOT zero on return!)
1563             *
1564 F49ED 774E GDIRSM  GOSUB  Getzer       Read 2 bytes=0, 2 more into C[A]
1565 F49F1 400         RTNC                Error if not zero
1566             *
1567             * Use D1 as a temporary holding area for multiplicand
1568             *
1569 F49F4 131         D1=A
1570 F49F7 D0          A=0    A            Clear product area
1571             * D1 is multiplicand, C[A] is multiplier, A[A] is zero
1572 F49F9 137         CD1EX
1573             * D1 is multiplier, C[A] is multiplicand, A[A] is zero
1574 F49FC 1C0 GDIRSm  D1=D1- 1            Decrement multiplier...
1575 F49FF 490         GOC    GDIRsM       ...End of loop!
1576 F4A02 CA          A=A+C  A            Add multiplicand to product...
1577 F4A04 57F         GONC   GDIRSm       If no carry, repeat loop!
1578 F4A07 02          RTNSC               If carry, WAY too big!
1579             *-
1580             *-
1581             *
1582             * Now product in A[A], multiplicand in C[A]
1583             *
1584 F4A09 24  GDIRsM  P=     4            ...point to high nibble...
1585 F4A0B 90C         ?A#0   P            ...and check if product too big.
1586 F4A0E 00          RTNYES              TOO big!
1587             *
1588             * Return with C[3:0] = multiplicand, A[3:0] = product
1589             *
1590 F4A10 03          RTNCC               Size is OK!
1591             ************************************************************
1592             ************************************************************
1593             **
1594             ** Name:       F->SCR - Write "FFF"s to SCRTCH ram
1595             **
1596             ** Category:   LOCAL
1597             **
1598             ** Purpose:
1599             **      Write 64 nibbles of "FFF" into SCRTCH RAM
1600             **
1601             ** Entry:
1602             **      None
1603             **
1604             ** Exit:
```

```
1605                **      Carry clear, D1 @ =SCRTCH+64,P=15
1606                **
1607                ** Calls:      D1=SCR
1608                **
1609                ** Uses.......
1610                **  Inclusive: C[W],D1,P,SCRTCH[63:0]
1611                **
1612                ** Stk lvls:   1 (D1=SCR)
1613                **
1614                ** History:
1615                **
1616                **    Date      Programmer            Modification
1617                **  --------    ----------    --------------------------------
1618                **  02/18/83      NZ          Added call to D1=SCR to pack code
1619                **  01/06/83      NZ          Added routine and documentation
1620                **
1621                ********************************************************************
1622                ********************************************************************
1623 F4A12 7610 =F->SCR GOSUB  D1=SCR
1624 F4A16 AF2         C=0     W
1625 F4A19 A7E         C=C-1   W               C="FFFFFFFFFFFFFFFF"
1626 F4A1C 23          P=      3               Write out 64 nibbles (4*16)
1627 F4A1E 1557 F->SC!  DAT1=C  W
1628 F4A22 17F         D1=D1+  16
1629 F4A25 0D          P=P-1                   Decrement counter
1630 F4A27 56F         GONC    F->SC!          Not done...continue
1631 F4A2A 03          RTNCC                   Done...carry clear!
1632                *-
1633                *-
1634 F4A2C 1F00 =D1=SCR D1=(5)  =SCRTCH
           000
1635 F4A33 01          RTN
1636                *-
1637                *-
1638 F4A35 20   DdlWrt  P=      =Write
1639 F4A37 8C00 Ddl     GOLONG  =DDL
           00
1640                *-
1641                *-
1642 F4A3D 20   =DdtRd  P=      =Read
1643 F4A3F 7F00         GOSUB   Ddt
1644 F4A43 400          RTNC
1645 F4A46 6358 Tstata  GOTO    TSTATA
1646                *-
1647                *-
1648 F4A4A 6C78 Seeka   GOTO    SEEKA
1649                *-
1650                *-
1651 F4A4E 6448 Tstat   GOTO    TSTAT
1652                *-
1653                *-
1654 F4A52 8C00 Ddt     GOLONG  =DDT
           00
1655                *-
1656                *-
```

```
1657 F4A58 8C00 Putc    GOLONG =PUTC
           00
1658            *-
1659            *-
1660 F4A5E 840  Writit  ST=0   =LoopOK       Do not abort out with ONE ATTN
1661 F4A61 8C00         GOLONG =WRITIT
           00
1662            *-
1663            *-
1664 F4A67 71CF Readsc  GOSUB  D1=SCR
1665 F4A6B 8C00 Readsu  GOLONG =READSU
           00
1666            *-
1667            *-
1668 F4A71 8C00 Putd    GOLONG =PUTD
           00
1669            *-
1670            *-
1671 F4A77 8C00 Putdx   GOLONG =PUTDX
           00
1672            *-
1673            *-
1674 F4A7D 8C00 Mtyl    GOLONG =MTYL
           00
1675            *-
1676            *-
1677 F4A83 8C00 Start   GOLONG =START
           00
1678            *-
1679            *-
1680 F4A89 816  Csrc5   CSRC
1681 F4A8C      Cslc12
1682 F4A8C 816  Csrc4   CSRC
1683 F4A8F 8C00 Csrc3   GOLONG =CSRC3
           00
1684            *-
1685            *-
1686 F4A95      Csrc10
1687 F4A95 812  Cslc6   CSLC
1688 F4A98 812  Cslc5   CSLC
1689 F4A9B      Csrc12
1690 F4A9B 812  Cslc4   CSLC
1691 F4A9E      Csrc13
1692 F4A9E 8C00 Cslc3   GOLONG =CSLC3
           00
1693            *-
1694            *-
1695 F4AA4 8C00 Asrc4   GOLONG =ASRC4
           00
1696            *-
1697            *-
1698 F4AAA 8D00 Ymdhms  GOVLNG =YMDHMS
           000
1699            ****************************************************************
1700            ****************************************************************
```

```
1701            **
1702            ** Name:       NXTENT - Move to next directory entry
1703            ** Name:       LSTENT - Move to previous directory entry
1704            **
1705            ** Category:   PILUTL
1706            **
1707            ** Purpose:
1708            **      Increment/decrement to next/last directory entry
1709            **
1710            ** Entry:
1711            **      C[3:0] is the current entry
1712            **
1713            ** Exit:
1714            **      C[3:0] is next/last entry
1715            **      P=0
1716            **      Carry set if crossed record boundary, else clear
1717            **
1718            ** Calls:      None
1719            **
1720            ** Uses.......
1721            **  Inclusive: C[3:0],P
1722            **
1723            ** Stk lvls:   0
1724            **
1725            ** History:
1726            **
1727            **    Date      Programmer              Modification
1728            **  --------    ----------    ---------------------------------
1729            **  12/08/82      NZ          Added routine and documentation
1730            **
1731            ************************************************************
1732            ************************************************************
1733 F4AB1 D9   =NXTEN+ C=B    A
1734 F4AB3 23   =NXTENT P=     3
1735 F4AB5 0B           CSTEX
1736 F4AB7 853          ST=1   3             Set high bit to propagate carry
1737 F4ABA 0B           CSTEX
1738 F4ABC B16          C=C+1  WP            Increment counter
1739 F4ABF 0B           CSTEX
1740 F4AC1 863          ?ST=0  3             Is this zero (Nibble is zero)?
1741 F4AC4 11           GOYES  LSTEN1        Yes...set carry
1742 F4AC6 5E0          GONC   LSTEN1        Go always...clear carry
1743            *-
1744            *-
1745 F4AC9 23   =LSTENT P=     3
1746 F4ACB A1E          C=C-1  WP
1747 F4ACE 0B           CSTEX
1748 F4AD0 873          ?ST=1  3             >7?
1749 F4AD3 20           GOYES  LSTEN1        Yes...set carry
1750 F4AD5 843  LSTEN1  ST=0   3             Clear unconditionally!
1751 F4AD8 0B           CSTEX
1752 F4ADA 20           P=     0             Always set P=0!!!
1753 F4ADC 01           RTN                  Carry set if new entry,else clear
1754            ************************************************************
1755            ************************************************************
```

```
1756            **
1757            ** Name:       NEWFIL,NEWFI+ - create a file on mass memory
1758            **
1759            ** Category:   FILUTL
1760            **
1761            ** Purpose:
1762            **      Create a new file on a medium, given a pointer to the
1763            **      file data and all info needed to create the directory
1764            **      entry. If NEWFIL is called by CREATE, the file will be
1765            **      initialized according to its create code.
1766            **
1767            ** Entry:
1768            **      ST[=sOVERW]=1 if overwrite existing file, 0 if error on
1769            **         existing file
1770            **      D[X] is device address (D[B]=0 if LOOP)
1771            **      R0 is first 8 chars of name
1772            **      R4[15:12] is last 2 chars of name
1773            **      R1[5:0] is new file size in bytes
1774            **      R1[9:6] is new file type
1775            **      R1[14:10] is new file data start (RAM address)
1776            **         (If zero, don't copy any file...check CCode)
1777            **      R1[15] = 0 if called by COPY with device spec,
1778            **         "F" if called by COPY with LOOP or non-mass storage
1779            **           device (D[B]#0 means non-mass storage device)
1780            **         create code if called by CREATE
1781            **      R2[7:0] is data for implementation bytes ([B] is first
1782            **         byte of implementation field...byte 28)
1783            **      (R2[B] is FIRST byte of implementation info)
1784            **   NEWFIL:
1785            **      D0 points to the mailbox
1786            **
1787            ** Exit:
1788            **      Carry clear:
1789            **        P=0, R3 is file information (B[W] internally):
1790            **          [3:0]: Current directory pointer (of no value)
1791            **          [7:4]: Pointer to start of data area for file
1792            **          [11:8]: Pointer to old directory location (if found)
1793            **          [15:12]: Pointer to new directory location of file
1794            **        R1 is unchanged from entry conditions
1795            **          (If R1[S]="F" and R1[B]#"00" then R1[5:2] has been
1796            **           incremented, R1[B]=0)
1797            **        The file has been created on the mass storage medium
1798            **      Carry set:
1799            **        Error (P,C[0] are error code)
1800            **
1801            ** Calls:      START,CHKBIT,GDIRST,SEEKA,DdtRd,READSC,GT2BYT,
1802            **             NXTENT,PT2BYT,YMDHMS,MTYL,<ENDTAP>,I/OFND,PURFIB,
1803            **             FTYPF#,CHKSEC,CHKSIZ,PUGFIB,NEWF80,NEWF84,NEWF90,
1804            **             NEWF.0,GETMBX,D1=SCR,F->SCR
1805            **             CSRC3;4;5;8;9;12,ASRC4,CSLC3;4;5;8;12
1806            **
1807            ** NEWF80 -->v ASRC4;8,CSRC2;3;12,CSLC3,YMDHMS,PT2BYT,DdlPwr,
1808            **             SEEKA,MTYL,DDL,PUTD,PUTC,D1=SCR
1809            ** NEWF84 -->v PT2BYT,CSLC2;6,MTYL,GT2BYT,CSRC13
1810            ** PUTDR# -->v SEEKA,MTYL
```

```
1811            ** NEWF90 -->v DdlPwr,DDL,PUTD
1812            ** PUTDIR ---> DDL,D1=SCR,<NEWF.3>
1813            **
1814            ** NEWF.0 -->v CSRC4;10,SEEKA,MTYL,DDL,<INITFL>
1815            ** NEWF.3 ---> WRITIT,GETST,PUTC,<TSTAT>
1816            **
1817            ** Uses.......
1818            **  Exclusive: A,B,C,D,R0,R2,R3,R4,D0,D1,P
1819            **  Inclusive: A,B,C,D,R0,R2,R3,R4,D0,D1,P,SCRTCH[63:0],ST[8,4:0]
1820            **
1821            ** Stk lvls:   5 (PUGFIB)(Only if deleting FIB entry:file existed
1822            ** Stk lvls:   4 (GDIRST)(NEWF80;YMDHMS)
1823            **
1824            ** Detail:
1825            **      Consolidates into one pass through the directory the
1826            **        following actions for mass storage:
1827            **             1. Find the file on the medium (if present)
1828            **             2. Find a space on the medium sufficient to hold
1829            **                  the file, giving preference to the place
1830            **                  it was before (if found in 1.)
1831            **             3. Purge the old directory entry, if not using
1832            **                  same entry for new file
1833            **             4. Write the new directory entry
1834            **             5. Copy the file to the data area of the medium
1835            **
1836            ** Algorithm:
1837            **   0: Get directory information
1838            **      Initialize PTRC,PTRD,PTRF,PTRL,PTRN,PFC
1839            **       (PTRC is current directory entry   <== dir_start
1840            **        PTRD is "hole" in directory space <== dir_start
1841            **        PTRF is "hole" in file space      <== 0
1842            **        PTRL is old directory entry       <== 0
1843            **        NEW is new directory entry flag   <== 0
1844            **        PFC is count of purged files      <== 0
1845            **        )
1846            **      Seek to the start of the directory space
1847            **      --
1848            **   1: Read a directory entry @ PTRC into =SCRTCH
1849            **      --
1850            **      -- Check if done with medium directory
1851            **      --
1852            **      IF ((end of directory) THEN 5:
1853            **      --
1854            **      -- Check if have enough information already
1855            **      --
1856            ** 1.2: IF (PTRL#0 AND NEW#0) THEN 5:
1857            **      --
1858            **      -- Check if in_file is purged
1859            **      --
1860            ** 1.3: IF (in_file_type = 0) THEN 2:
1861            **      --
1862            **      -- Check if names match (found old file)
1863            **      --
1864            **      IF (in_file_name # new_file_name) THEN 3:
1865            **      --
```

```
1866          **      -- Check if overwrite is permitted
1867          **      --
1868          **      IF (ST[sOVERW]=0) THEN ERROR (File Exists)
1869          **      --
1870          **      IF (old file is secure) THEN ERROR (File protect)
1871          **      --
1872          **      Mark FIB entry to be purged if old file is open
1873          **      --
1874          **      -- Check if room for new file in old file
1875          **      --
1876          **      IF (in_file_space < new_file_size) THEN 1.5:
1877          **      --
1878          **      -- It fits here...use this entry!
1879          **      --
1880          **      PTRF <== in_file_start
1881          **      PTRD <== PTRC
1882          **      --
1883          **      Write new_file_implementation into SCRTCH directory entry
1884          **      Write new_file_type into SCRTCH directory entry
1885          **      --
1886          **      Get current time and date from mainframe
1887          **      --
1888          **      GOSUB 8.4: -- Write time&date, output entry @ PTRD
1889          **      --
1890          **      GOTO 7:    -- Transfer file data to PTRF, exit cleanly
1891          **      ------------------------------------------------------
1892          **      --
1893          **      -- Found old file, file won't fit here...mark as purged
1894          **      --
1895          ** 1.5: PTRL <== PTRC
1896          **      --
1897          **      -- Count a purged file, get the next directory entry
1898          **      --
1899          **   2: PFC <== PFC + 1
1900          **      GOTO 4:
1901          **      ------------------------------------------------------
1902          **   3: --
1903          **      -- Names don't match...check if found new space yet
1904          **      -- (If found new space, continue to look for old name)
1905          **      --
1906          **      IF (NEW#0) THEN 4:
1907          **      --
1908          **      -- Check if this file terminates a purged block AND
1909          **      -- the file would fit here
1910          **      --
1911          **      IF (PFC#0 AND ((in_file_start - PTRF)>=new_file_size))
1912          **                              THEN NEW <== 1 @ GOTO 4:
1913          **      --
1914          **      -- Won't fit OR not termination of purged block
1915          **      --
1916          ** 3.4: PFC <== 0
1917          **      PTRF <== in_file_start + in_file_length
1918          **      PTRD <== PTRC + 1
1919          **      --
1920          **      -- Fall through to code to loop back for next entry
```

```
1921            **      --
1922            **   4: PTRC <== PTRC + 1
1923            **      IF (NOT End_of_directory) THEN 1:
1924            **      --
1925            **      PhEOD <== 1  -- Set Physical End of directory flag...
1926            **      --
1927            **      -- ...and fall through to End_of_directory code
1928            **      --
1929            **   5: --
1930            **      -- Check why we are done...end_of_file or finished
1931            **      --
1932            **      IF (NEW=0) THEN 6:
1933            **      --
1934            ** 5.5: GOSUB 8:   -- Purge the old file, create new directory
1935            **      --
1936            **      GOTO 7:    -- Copy the data to (PTRF), exit cleanly
1937            **      ------------------------------------------------------
1938            **   6: --
1939            **      -- Check physical end_of_directory and no purged files
1940            **      --
1941            **      IF (PhEOD AND PFC=0) THEN ERROR ("Directory Full")
1942            **      --
1943            **      -- Check if room at end of medium for new_file
1944            **      --
1945            ** 6.2: IF (NOT (room at end)) THEN ERROR ("End of Medium")
1946            **      --
1947            **      GOSUB 8:   -- Purge the old file, create new directory
1948            **      --
1949            **      -- Check if room for logical end_of_directory mark
1950            **      --
1951            **      IF ((PTRD + 1) = physical_end_of_directory) THEN GOTO 7:
1952            **      --
1953            **      -- Write an End of Directory mark to the medium
1954            **      --
1955            ** 6.7: Set SCRTCH="FF...FF" -- Set up EOD mark in RAM
1956            **      --
1957            **      GOSUB 9:   -- Write the directory entry here
1958            **      --
1959            **   7: GOSUB #:   -- Copy the data to the medium
1960            **      --
1961            **      -- Delete the FIB entry marked to be deleted (if any)
1962            **      --
1963            **      GOSUB Purge_marked_FIB_entry (PUGFIB)
1964            **      --
1965            **      IF (device is mass storage) THEN GOTO Rewind&status
1966            **      --
1967            **      -- Destination is LOOP/non-mass storage
1968            **      --
1969            ** 7.5: IF (device is LOOP) THEN GOTO send ETO
1970            **      --
1971            **      GOTO untalk_unlisten_end
1972            **      ------------------------------------------------------
1973            **      ------------------------------------------------------
1974            **      --
1975            **      -- Subroutine to write the directory to the medium
```

```
1976            **      --
1977            **      -- Found room for the new file...check if found old
1978            **      -- (If found it and writing somewhere else, purge it)
1979            **      --
1980            **   8: IF (PTRL#0 AND PTRL#PTRD) THEN PTRL_file_type <== 0
1981            **      --
1982            **      -- Before copying data, build the new directory entry
1983            **      --
1984            ** 8.2: Get current time and date: set up type, start addr,
1985            **         and length of file
1986            **      --
1987            ** 8.4: Set up time and date, volume #, end flag, and implementat
1988            **      --
1989            **      -- Now directory entry is set up...write it to the medium
1990            **      --
1991            **      GOSUB SEEK(PTRD)
1992            **      --
1993            **      -- Write the new directory entry to the medium
1994            **      --
1995            **   9: Set up partial write mode to read in the record, repositi
1996            **      --
1997            ** 9.5: Set to write mode (buffer 0 contains the record)
1998            **      Set the byte pointer to the correct entry
1999            **      --
2000            **      Write the new entry
2001            **      --
2002            **      RETURN -- End of subroutine 8:
2003            **      ------------------------------------------------------
2004            **      ------------------------------------------------------
2005            **      --
2006            **      -- Subroutine to write the data to the medium
2007            **      --
2008            **   #: IF [(data length=0) OR (data address=0) OR "LOOP"] AND
2009            **         (this is a COPY)) THEN RETURN
2010            **      --
2011            **      -- If this is a COPY, transfer data else initialize it
2012            **      --
2013            **      IF (NOT LOOP) THEN SEEK(PTRF)
2014            **      --
2015            **      IF CREATE THEN initialize data area (INITFL), RETURN
2016            **      --
2017            **      COPY new_file_data TO (PTRF) (Send last byte as END)
2018            **      --
2019            **      RETURN -- End of subroutine #:
2020            **
2021            **
2022            ** History:
2023            **
2024            **    Date      Programmer            Modification
2025            **   --------   ----------   --------------------------------
2026            **  10/11/83       NZ        Updated documentation
2027            **  09/01/83       NZ        Added call to DELFIB to fix bug
2028            **                           with not closing assign # to the
2029            **                           destination of a COPY command,
2030            **                           packed to install this fix
```

```
2031            **  07/18/83      NZ          Added status bit for overwriting
2032            **                              file
2033            **  05/12/83      NZ          Changed CHKMAS call to use bits
2034            **                              that are set by FILSPx
2035            **  03/02/83      NZ          Added sending "ENDM to Diamond
2036            **  02/05/83      NZ          Added CHKMAS in NEWFI+
2037            **  02/04/83      NZ          Added LOOP check in several spots
2038            **  02/03/83      NZ          Rearranged order of copy...now
2039            **                              writes directory entry BEFORE
2040            **                              writing the data
2041            **  11/19/82      NZ          Added documentation
2042            **
2043            ****************************************************************
2044            ****************************************************************
2045 F4ADE 71AF =NEWFI+ GOSUB  Start         Set up the loop
2046 F4AE2 400         RTNC                 Error???                        <<<<
2047            *
2048            * Now check if mass storage device...if not, check R1[S]:
2049            *       If R1[S]=0, set R1[S]="F" (not mass storage)
2050            *       If R1[S]#0, this is a create...error!
2051            *
2052 F4AE5 96B         ?D=0   B             "LOOP"?
2053 F4AE8 90          GOYES  NEWF++        Yes...set R1[S]
2054            *
2055            * Check if bit "4" of D[3] is set...if so, then mass storage
2056            *
2057 F4AEA 7028        GOSUB  CHKBIT        Check mass storage bit
2058 F4AEE 4B0         GOC    NEWFIL        Mass storage...continue on!
2059 F4AF1 119  NEWF++ C=R1
2060 F4AF4 A4E         C=C-1  S
2061 F4AF7 109         R1=C                 Set for "LOOP" or not MS
2062 F4AFA 119  =NEWFIL C=R1                Check if LOOP or Non-MS device
2063 F4AFD B46         C=C+1  S
2064 F4B00 590         GONC   NEWF01        Not LOOP
2065 F4B03 AF1         B=0    W             LOOP...set all pointers=0, enter
2066 F4B06 6082        GOTO   NEWF55          at a later entry point
2067            *-
2068            *-
2069 F4B0A 7ACD NEWF01 GOSUB  GDIRST        Get directory start, etc
2070            *
2071            * 0: Initialization
2072            *
2073            * GDIRST leaves start of directory (Dstrt) in A[X], length of
2074            * directory (Dleng) in A[6:3], address of next record after
2075            * the last one on the medium (Tlast) in A[10:7]
2076            *
2077            * Now initialize my internal pointers
2078            *
2079            * Name  Value  Register nibs Description:
2080            * ----  -----  -------- ---- --------------------------------
2081            * PTRC: Dstrt  B[3:0]     4  Current directory pointer
2082            * PTRD: Dstrt  B[15:12]   4  Directory pointer (new space)
2083            * PTRF: Dend   B[7:4]     4  File pointer (to data area)
2084            * PTRL:   0    B[11:8]    4  Pointer to old name (Last) entry
2085            * NEW:    0    D[S]       1  Flag- indicates PTRD is new entry
```

```
2086            * PFC:    0    D[14]      1  Purged file currently found
2087            * PhEOD:  0    D[13]      1  Physical end_of_directory reached
2088            * NName:given  R0,R4[15:12]  New file name (20 nibbles)
2089            * NSize:given  R1[5:0]    6  New file size (bytes)
2090            * NType:given  R1[9:6]    4  New file type
2091            * NData:given  R1[14:10]  5  New file data start (in RAM)
2092            * CCode:given  R1[15]     1  Create code (if not zero,F)
2093            * NImpl:given  R2[7:0]    8  New file implementation bytes
2094            * Dlenl:Dleng  D[8:5]     4  Directory records left to process
2095            *                              (includes current record)
2096            * Tendr:Tlast  D[12:9]    4  Medium end (address of next record)
2097            *
2098            * All directory pointers are of the form [3 nibs][1 nib];
2099            *    The [3 nibs] field is the directory record number.
2100            *    The [1 nib] field is the entry number within the record.
2101            *
2102            * If carry, check what the error is...if "New Medium", try again
2103            *
2104 F4B0E 5A1          GONC   NEWF05        OK...continue
2105            *
2106            * Check for "New Medium" error ...close files, continue
2107            *
2108 F4B11 880          ?P#    =eTAPE
2109 F4B14 00           RTNYES               Error during status
2110 F4B16 80F0         CPEX   0
2111 F4B1A 880          ?P#    =eNEWTA       New medium?
2112 F4B1D 20           GOYES  NEWF03
2113 F4B1F 80F0 NEWF03  CPEX   0            Carry: not "New Medium"
2114 F4B23 400          RTNC                 If carry, return the error
2115 F4B26 53E          GONC   NEWF01        Go always...try again!
2116            *-
2117            *-
2118            *
2119            * Seek the first record of the directory...in A[X]
2120            *
2121 F4B29 AD0  NEWF05  A=0    M             Clear high nibbles
2122 F4B2C 7A1F         GOSUB  Seeka         Seek to that record
2123 F4B30 400          RTNC                 Error with medium or loop
2124 F4B33 760F         GOSUB  DdtRd         Read command
2125 F4B37 400          RTNC                 Error
2126            *
2127            * 1: Read in an entry (at PTRC)
2128            *
2129 F4B3A 20   NEWF10  P=     0
2130 F4B3C 3500         LC(6)  (=mSDA)+32    Read 32 bytes...
           0000
2131 F4B44 7F1F         GOSUB  Readsc        ...into =SCRTCH!
2132 F4B48 400          RTNC                 Error!
2133 F4B4B 1D00         D1=(2) (=SCRTCH)+20  Type!
2134 F4B4F 15F3         C=DAT1 4
2135 F4B53 23           P=     3
2136 F4B55 B16          C=C+1  WP            If carry, then End of directory
2137 F4B58 560          GONC   NEWF12        Not end of directory
2138 F4B5B 6622 NEWF11  GOTO   NEWF50        End of directory!
2139            *-
```

```
2140                *-
2141 F4B5F 94B  NEWF12  ?D=0   S             Is NEW=0?
2142 F4B62 B1           GOYES  NEWF13        Yes...continue
2143 F4B64 AF9          C=B    W
2144 F4B67 8E00         GOSUBL =CSRC8        Get PTRL into C[3:0]
           00
2145 F4B6D 91A          ?C=0   WP            Is PTRL=0? (P is 3)
2146 F4B70 D0           GOYES  NEWF13        Yes...continue
2147                *
2148                * PTRL#0 and NEW#0...call it end of directory
2149                *
2150 F4B72 58E          GONC   NEWF11        Go always!
2151                *-
2152                *-
2153 F4B75 6751 NEWF2.  GOTO   NEWF20        Jump (out of range)
2154                *-
2155                *-
2156 F4B79 6F51 NEWF3.  GOTO   NEWF30        Jump (out of range)
2157                *-
2158                *-
2159 F4B7D      NEWF13
2160                *
2161                * Check if in_type=0
2162                *
2163 F4B7D 15F3         C=DAT1 4             Reread type from SCRTCH+#20
2164 F4B81 91A          ?C=0   WP            Purged file?
2165 F4B84 1F           GOYES  NEWF2.        Yes...process it
2166                *
2167                * This is not a purged file...check if names match
2168                *
2169 F4B86 1C3          D1=D1- 4             Set D1<-last 2 characters of name
2170 F4B89 15B3         A=DAT1 4             Read them into A[3:0] for now
2171 F4B8D 1CF          D1=D1- 16            First 8 characters of name
2172 F4B90 1577         C=DAT1 W             Read them!
2173 F4B94 120          AROEX                First 8 chars in A[W], C[W]
2174 F4B97 976          ?A#C   W
2175 F4B9A 20           GOYES  NEWF14        Sets carry if no match
2176 F4B9C 120  NEWF14  AROEX                Swap name back into R0, A[3:0]
2177 F4B9F 49D          GOC    NEWF3.        Not a match...continue
2178                *
2179                * First 8 chars match...check if last 2 also match
2180                *
2181 F4BA2 11C          C=R4
2182 F4BA5 72FE         GOSUB  Csrc12        Get last 2 chars in C[3:0]
2183 F4BA9 912          ?A=C   WP
2184 F4BAC 50           GOYES  NEWF1a        Match...check room
2185 F4BAE 5AC          GONC   NEWF3.        Go always...Not match
2186                *-
2187                *-
2188                *
2189                * Names match...check if overwrite permitted (if not, error)
2190                *
2191 F4BB1 20   NEWF1a  P=     0
2192 F4BB3 300          LC(1)  =eEFILE       File exists
2193 F4BB6 860          ?ST=0  =sOVERW       Overwrite it?
```

```
2194 F4BB9 92          GOYES  NEWF1d        No...error (Duplicate file)
2195           *
2196           * Overwrite permitted...check if file is secure
2197           *
2198 F4BBB 1D00        D1=(2) (=SCRTCH)+20  Point to type field
2199 F4BBF AF9         C=B    W             Save B in R3 temporarily!
2200 F4BC2 10B         R3=C
2201           *
2202           * FTYPF# destroys R0, but the name is also in SCRTCH[15:0]
2203           *
2204 F4BC5 7925        GOSUB  GT2BY0        Read file type
2205 F4BC9 DA          A=C    A             File type is in A[A] now
2206 F4BCB 8E00        GOSUBL =fTYPF#       Get file type #
          00
2207 F4BD1 541         GONC   NEWF1c        Not found...OK (continue)
2208           *
2209           * Found...check if secure
2210           *
2211 F4BD4 8E00        GOSUBL =CHKSEC       If secure, returns with carry set
          00
2212 F4BDA 5B0         GONC   NEWF1c        Not secure...OK to continue
2213           *
2214           * File is secure...error!
2215           *
2216 F4BDD 20  =fPROT  P=     0             Set up "File Protected" error
2217 F4BDF 300         LC(1)  =efPROT       File Protected!
2218 F4BE2 20  NEWF1d  P=     =eTAPE
2219 F4BE4 02          RTNSC
2220           *-
2221           *-
2222           *
2223           * Not secure...kill the FIB entry (if any) for the file!
2224           *
2225 F4BE6      NEWF1c
2226           *
2227           * First build the FIB file data pointer
2228           *
2229 F4BE6 DB            C=D    A
2230 F4BE8 1F00          D1=(5) (=SCRTCH)+28  Start address (third byte)
          000
2231 F4BEF 7105        GOSUB  GT2BYT        Read two bytes!
2232 F4BF3 77AE        GOSUB  Cslc3
2233 F4BF7 3200        LC(3)  =bFIB
          0
2234 F4BFC 8E00        GOSUBL =i/OFND       Find the FIB buffer
          00
2235 F4C02 798E        GOSUB  Csrc3         Now C[6:0] is pointer!
2236           *
2237           * D1 @ FIB buffer, C[6:0] is address of file
2238           *
2239 F4C06 8E00        GOSUBL =PURFIB       Find and mark the FIB entry
          00
2240           *
2241           * Restore R0 from SCRTCH and B[W] from R3
2242           *
```

```
2243 F4C0C 7C1E         GOSUB  D1=SCR
2244 F4C10 1577         C=DAT1 W
2245 F4C14 108          R0=C                    Restore R0[W] from SCRTCH
2246 F4C17 11B          C=R3
2247 F4C1A AF5          B=C    W                Restore B[W] from R3
2248             *
2249             * Registers are restored...check if room for new file here
2250             *
2251 F4C1D 111          A=R1                    Get file length (given)
2252 F4C20 1D00         D1=(2) (=SCRTCH)+36     Length of file field @ 3rd byte
2253             *
2254             * NOTE: if length of existing file is > 2^16 sectors, this
2255             * code will treat it as if it were (size modulo 2^16)
2256             *
2257 F4C24 7CC4         GOSUB  GT2BYT           ...Read 2 bytes, start at D1
2258 F4C28 F2           CSL    A
2259 F4C2A 25           P=     5
2260 F4C2C B92          CSL    WP               Now C[5:0] is length in bytes
2261 F4C2F 99A          ?A<=C  WP               Does it fit?
2262 F4C32 60           GOYES  NEWF1b           Yes...set it up!
2263 F4C34 6680         GOTO   NEWF15           No...continue
2264             *-
2265             *-
2266             *
2267             * New file will fit in space for the old file (already on medium)
2268             *
2269             * Copy start address to PTRF
2270             *
2271 F4C38 1CB  NEWF1b  D1=D1- 12               Point to start address @ 3rd byte
2272 F4C3B 75B4         GOSUB  GT2BYT           Read 2 bytes into C[3:0]
2273 F4C3F 785E         GOSUB  Cslc4            ...shift to C[7:4]...
2274 F4C43 23           P=     3
2275 F4C45 A99          C=B    WP               ...copy PTRC to C[3:0]...
2276 F4C48 27           P=     7
2277 F4C4A A95          B=C    WP               ...and set PTRF<==in_start
2278 F4C4D 7B3E         GOSUB  Csrc4            Shift PTRC to C[15:12]
2279 F4C51 2B           P=     11
2280 F4C53 A99          C=B    WP
2281 F4C56 10B          R3=C                    Copy PTRC==>PTRD! (save B in R3)
2282 F4C59 112          A=R2                    Get implementation bytes into A
2283 F4C5C 1D00         D1=(2) (=SCRTCH)+56     (Implementation bytes)
2284 F4C60 1597         DAT1=A 8                Write out the 4 bytes!
2285             *
2286             * Update the file type to the "new" type
2287             *
2288 F4C64 119          C=R1                    Get type from R1[9:6]
2289 F4C67 712E         GOSUB  Csrc4            Get to C[5:2]
2290 F4C6B 1D00         D1=(2) (=SCRTCH)+20     Point to type field
2291 F4C6F 7894         GOSUB  PT2BYT           Output 2 bytes from C[5:2] to D1
2292             *
2293             * Now B[W] in R3; Save R1[W] in R2; D[A] in R4[9:5]
2294             * (YMDHMS uses A-D,P,D0,D1,R0,R1,ST[7:0])
2295             *
2296 F4C73 119          C=R1
2297 F4C76 10A          R2=C                    R1 in R2
```

```
2298 F4C79 DB          C=D    A
2299 F4C7B 791E        GOSUB  Cslc5
2300 F4C7F 10C         R4=C                  D[A] in R4[9:5]
2301 F4C82 742E        GOSUB  Ymdhms
2302           *
2303           * Now C[11:0] is date info
2304           *
2305 F4C86 AF5         B=C    W              Save date info in B temporarily
2306 F4C89 11C         C=R4
2307 F4C8C 79FD        GOSUB  Csrc5
2308 F4C90 D7          D=C    A              Restore D[A]
2309 F4C92 7000        GOSUB  =Getmbx        Restore D0
2310 F4C96 1F00        D1=(5) (=SCRTCH)+56
        000
2311 F4C9D 15F7        C=DAT1 8              Recall impl bytes (for NEWF84)
2312 F4CA1 12A         CR2EX                 Restore R2, fetch R1 value
2313 F4CA4 109         R1=C                  Restore R1
2314 F4CA7 11B         C=R3                  Recall B[W] value
2315 F4CAA AFD         BCEX   W              Restore B[W], fetch date info
2316 F4CAD 1C2         D1=D1- 3              Position to where NEWF84 expects
2317 F4CB0 7EF2        GOSUB  NEWF84         Write the date, vol label, impl
2318 F4CB4 400         RTNC                  Error somewhere!
2319 F4CB7 6E61        GOTO   NEWF70         Copy file to (PTRF), exit cleanup
2320           *-
2321           *-
2322           *
2323           * 1.5: Found the old file, new file won't fit there
2324           *
2325 F4CBB      NEWF15
2326           *
2327           * Found old file, but it's too small now...consider it purged
2328           *
2329 F4CBB D9          C=B    A              Set PTRL<==PTRC
2330 F4CBD 8E00        GOSUBL =CSLC8
        00
2331 F4CC3 27          P=     7
2332 F4CC5 A99         C=B    WP
2333 F4CC8 2B          P=     11
2334 F4CCA A95         B=C    WP             Now PTRL=PTRC
2335           *
2336           * 2: Mark a purged file and loop back
2337           *
2338 F4CCD 2E    NEWF20  P=     14
2339 F4CCF B07   NEWF25  D=D+1  P              Make PFC non-zero
2340 F4CD2 4CF         GOC    NEWF25         If carry, wrap around!
2341 F4CD5 6080 NEWF4.  GOTO   NEWF40         Increment to next entry, loop back
2342           *-
2343           *-
2344           *
2345           * 3: Names don't match, non-purged file...check if found a
2346           *    place for the file yet (if so, continue looking for the
2347           *    old file on the medium)
2348           *
2349 F4CD9 94F   NEWF30  ?D#0   S              Is NEW#0?
2350 F4CDC 9F          GOYES  NEWF4.         Yes...continue looking for old
```

```
2351            *
2352            * Check if (PFC#0) AND ((Start-PTRF) >= new_size)
2353            *
2354            * First check PFC=0 (If zero, skip)
2355            *
2356 F4CDE 2E           P=     14
2357 F4CE0 90B          ?D=0   P              Is PFC zero?
2358 F4CE3 D3           GOYES  NEWF34         Yes...reset PTRF, PTRD and cont
2359            *
2360            * Now check if enough room!
2361            *
2362 F4CE5 1D00         D1=(2) (=SCRTCH)+28   In_file_start (Start@ third byte)
2363 F4CE9 7704         GOSUB  GT2BYT         Read 2 bytes,start @ D1,to C[3:0]
2364            *
2365            * C[A] is now In_file_start...get PTRF, check if file fits.
2366            *
2367 F4CED 7900         GOSUB  CHKSIZ         Check if fits (carry if not)
2368 F4CF1 4E2          GOC    NEWF34         Doesn't fit...continue
2369            *
2370            * The new file WILL fit at PTRF
2371            *
2372 F4CF4 B47          D=D+1  S              NEW <== 1 (PTRD is location)
2373 F4CF7 5E5          GONC   NEWF40         Go always
2374            *-
2375            *-
2376 F4CFA AF4  CHKSIZ  A=B    W              Get PTRF into A[3:0]
2377 F4CFD 73AD         GOSUB  Asrc4
2378 F4D01 23           P=     3
2379 F4D03 B12          C=C-A  WP             Compute (In_file_start - PTRF)
2380            *
2381            * Get NSize next, convert to records (Use next integer record)
2382            *
2383 F4D06 111          A=R1                  A[5:0] is size in bytes
2384 F4D09 822          SB=0                  Use the Sticky Bit to check if
2385 F4D0C BF4          ASR    W                any bits were shifted off the
2386 F4D0F BF4          ASR    W                end of A!
2387 F4D12 832          ?SB=0                 Any bits lost?
2388 F4D15 40           GOYES  CHKSIz         No...skip increment statement
2389            *
2390            * NOTE: if file size is ever > #FFFF00, this won't work
2391            *
2392 F4D17 E4           A=A+1  A              Increment A[3:0]
2393            *
2394            * Now C[3:0] is (In_file_start - PTRF), A[3:0] is NSize(Recs)
2395            *
2396 F4D19 996  CHKSIz  ?A>C   WP             Does it fit?
2397 F4D1C 00           RTNYES                No...set carry
2398 F4D1E 03           RTNCC                 Yes...clear carry
2399            *-
2400            *-
2401 F4D20      NEWF34
2402            *
2403            * File won't fit OR no purged files before it
2404            *
2405 F4D20 2E           P=     14
```

```
2406 F4D22 A83          D=0    P             PFC <== 0
2407               *
2408               * Set PTRF <== In_file_start + In_file_length
2409               *
2410 F4D25 1D00         D1=(2) (=SCRTCH)+28  Back to In_file_start...
2411 F4D29 77C3         GOSUB  GT2BYT        Read In_file_start into C[3:0]
2412 F4D2D DA           A=C    A             Save In_file_start in A[3:0]
2413 F4D2F 173          D1=D1+ 4             Move to In_file_length + 4
2414 F4D32 7EB3         GOSUB  GT2BYT        Read In_file_length into C[3:0]
2415               *
2416               * Now A[3:0] is In_file_start, C[3:0] is In_file_length
2417               *
2418 F4D36 23           P=     3             Set up for C=B WP below
2419               *
2420               * NOTE: if in_file(start+length)>#FFFF, this will be incorrect!
2421               *
2422 F4D38 C2           C=C+A  A             C[3:0] is in_file(start + length)
2423 F4D3A 7D5D         GOSUB  Cslc4         Shift to C[7:4]
2424 F4D3E A99          C=B    WP
2425 F4D41 27           P=     7
2426 F4D43 A95          B=C    WP            Copy to B[7:4]!
2427               *
2428               * Now set PTRD <== PTRC + 1
2429               * (PTRC is in C[3:0] NOW!)
2430               *
2431 F4D46 796D         GOSUB  NXTENT        Increment to next entry
2432 F4D4A 7E3D         GOSUB  Csrc4         Now C[3:0] is PTRC+1...
2433 F4D4E 2B           P=     11            ...move to C[15:11]...
2434 F4D50 A99          C=B    WP
2435 F4D53 AF5          B=C    W             ...and copy to PTRD!
2436               *
2437               * Fall through to...
2438               *
2439               * 4: Code to loop back for next entry
2440               *
2441 F4D56         NEWF40
2442               *
2443               * Increment PTRC, loop back if not record carry...else check
2444               * for end-of-directory, decrement record count
2445               *
2446 F4D56 775D         GOSUB  NXTEN+        C=PTRC, Increment to next entry
2447 F4D5A D5           B=C    A             Store back in PTRC
2448 F4D5C 460          GOC    NEWF45        Wrap!...Decrement record count
2449 F4D5F 6ADD NEWF1.  GOTO   NEWF10        Loop back for next entry
2450               *-
2451               *-
2452               *
2453               * Check for physical end of directory
2454               *
2455 F4D63 AFB  NEWF45  C=D    W
2456 F4D66 7F1D         GOSUB  Csrc5         Get Dlen1 into C[3:0]
2457               *
2458               * By the definition of Dlen1, this can't borrow (I check zero
2459               * every time I decrement and original value is > 0)
2460               *
```

```
2461 F4D6A CE          C=C-1  A          Decrement C[3:0] (Can't borrow)
2462 F4D6C 23          P=     3          Check C[3:0]
2463 F4D6E 91A         ?C=0   WP         Done?
2464 F4D71 C0          GOYES  NEWF48     Yes...Physical end of directory
2465 F4D73 712D        GOSUB  Cslc5
2466 F4D77 AF7         D=C    W          Store back into Dlen1
2467 F4D7A 54E         GONC   NEWF1.     Go always
2468            *-
2469            *-
2470 F4D7D 2D  NEWF48  P=     13         Point to PhEOD...
2471 F4D7F B07         D=D+1  P          ...and set it true
2472            *
2473            * 5: Reached end of file...process it now
2474            *
2475 F4D82     NEWF50
2476            *
2477            * First check if have a space for the new file
2478            *
2479 F4D82 94B         ?D=0   S          NEW=0?
2480 F4D85 D0          GOYES  NEWF60     Yes...no room yet
2481            *
2482            * Have room for it...process it here
2483            *
2484 F4D87 72E0 NEWF55  GOSUB  NEWF80     Purge old, create new file entry
2485 F4D8B 400         RTNC              Error during write
2486 F4D8E 6790        GOTO   NEWF70     Copy data to (PTRF), cleanup&exit
2487            *-
2488            *-
2489            *
2490            * 6: End of directory, no space found for file yet
2491            *
2492 F4D92     NEWF60
2493            *
2494            * If (PFC=0 AND physical End_of_directory) THEN Error!
2495            *
2496            * Check PFC=0 first
2497            *
2498 F4D92 2E          P=     14
2499 F4D94 90F         ?D#0   P
2500 F4D97 A1          GOYES  NEWF62     Need to check if room on medium
2501            *
2502            * Now check Physical End_of_directory
2503            *
2504 F4D99 2D          P=     13         Point to PhEOD...
2505 F4D9B 90B         ?D=0   P          ...check if reached PhEOD
2506 F4D9E 31          GOYES  NEWF62     Not physical end_of_directory
2507 F4DA0 20          P=     0          Is physical end_of_directory...
2508 F4DA2 300         LC(1)  =eDIRFL    Directory is full!
2509 F4DA5 20  NEWFeT  P=     =eTAPE     (Medium error)
2510 F4DA7 02          RTNSC
2511            *-
2512            *-
2513 F4DA9 20  NEWF61  P=     0
2514 F4DAB 300         LC(1)  =eEOTAP    End of medium
2515 F4DAE 46F         GOC    NEWFeT     Go always
```

```
2516                *-
2517                *-
2518                *
2519                * Not physical end_of_directory...check if room for file @ end
2520                *
2521 F4DB1          NEWF62
2522                *
2523                * PTRD points to the directory entry to be used...if room!
2524                *
2525                * First check if room at end of medium for this file.
2526                *
2527                * IF ((Tendr - PTRF) >= NSize) THEN room at end
2528                *
2529                * Get Tendr first...
2530                *
2531 F4DB1 AFB              C=D    W
2532 F4DB4 8E00             GOSUBL =CSRC9          Shift into C[3:0]
       00
2533                *
2534                * Now check if the file will fit here
2535                *
2536 F4DBA 7C3F             GOSUB  CHKSIZ          Check if room for file
2537 F4DBE 4AE              GOC    NEWF61          No...End of medium error
2538                *
2539                * Room for the file...write it here!
2540                *
2541 F4DC1 78A0             GOSUB  NEWF80          Purge old, create new dir entry
2542 F4DC5 400              RTNC                   Error during write
2543                *
2544                * Check if room for the end_of_directory mark here
2545                *
2546                * If got here by logical end_of_directory and PTRC is at the
2547                * last directory entry before physical EOD, then set PhEOD for
2548                * the following test!
2549                *
2550 F4DC8 75EC             GOSUB  NXTEN+          Increment to next entry
2551 F4DCC 562            GONC   NEWF67          Not new record...continue on
2552                *
2553                * New record...check if this was the LAST one
2554                *
2555 F4DCF AFB              C=D    W
2556 F4DD2 73BC             GOSUB  Csrc5           Get Dlenl into C[3:0]
2557 F4DD6 CE             C=C-1  A               Can't carry by its definition
2558 F4DD8 23             P=     3
2559 F4DDA 91A            ?C=0   WP              Physical end of directory?
2560 F4DDD 90             GOYES  NEWF66          Yes...no more records in directory
2561                *
2562                * If physical end_of_directory is false, then there IS room
2563                * for the end_of_directory mark.  If physical, then check if
2564                * PFC>1...if so, room for end_of_directory mark.
2565                *
2566                * Check first for physical end_of_directory
2567                *
2568 F4DDF 2D             P=     13
2569 F4DE1 90B            ?D=0   P               Is this physical EOD?
```

```
2570 F4DE4 F0          GOYES  NEWF67       No...OK to write EOD mark
2571 F4DE6      NEWF66
2572            *
2573            * Have reached physical end of directory...check if any purged
2574            * directory entries available to write the logical EOD mark
2575            *
2576 F4DE6 2E          P=     14           Check # of purged files
2577 F4DE8 A0F         D=D-1  P            (Decrement PFC)
2578 F4DEB 4A3         GOC    NEWF70       If PFC was zero, no room for EOD
2579 F4DEE 90B         ?D=0   P            More than one purged entry?
2580 F4DF1 53          GOYES  NEWF70       No...no room for EOD mark
2581            *
2582            * Write the end_of_directory mark
2583            *
2584 F4DF3 AF9  NEWF67  C=B    W            Get PTRD into C[15:12]...
2585 F4DF6 71AC        GOSUB  Csrc12       ...Move to C[3:0]...
2586 F4DFA 75BC        GOSUB  NXTENT       ...increment to next entry!
2587 F4DFE DA          A=C    A            Copy the pointer to A[3:0]
2588 F4E00 788C        GOSUB  Cslc12       ...move back to C[15:12]...
2589 F4E04 AF5         B=C    W            ..and copy back to B (Rest is OK)
2590 F4E07 814         ASRC                Entry # in A[S], record in A[X]
2591 F4E0A AD0         A=0    M            (Clear unused nibbles)
2592 F4E0D 793C        GOSUB  Seeka        Go to that record
2593 F4E11 400         RTNC                Error during seek
2594 F4E14 756C        GOSUB  Mtyl         I send data to the medium
2595 F4E18 400         RTNC                Error
2596            *
2597            * Write "FFF"s to SCRTCH (For the end_of_directory mark)
2598            *
2599 F4E1B 73FB        GOSUB  F->SCR       Write 64 nibs of "F" to SCRTCH
2600 F4E1F 77F1        GOSUB  NEWF90       Read the record, update, write
2601 F4E23 400         RTNC                If carry, error writing EOD
2602            *
2603            * 7: Copy the data to the medium
2604            *
2605 F4E26 7532 NEWF70  GOSUB  NEWF.0       Copy the data to the medium
2606 F4E2A 400         RTNC
2607            *
2608            * Fall into clean-up code...(rewind device, etc)
2609            *
2610 F4E2D 20          P=     0
2611 F4E2F AF9         C=B    W
2612 F4E32 10B         R3=C                Put B[W] into R3!
2613            *
2614            * Now delete the FIB buffer marked by PURFIB (if any)
2615            *
2616 F4E35 DB          C=D    A
2617 F4E37 10A         R2=C                Save D[A] in R2
2618 F4E3A 8F00        GOSBVL =PUGFIB      Delete first FIB marked as purged
          000
2619 F4E41 7000        GOSUB  =Getmbx      Get D0 back to the mailbox
2620 F4E45 11A         C=R2
2621 F4E48 D7          D=C    A
2622 F4E4A 11B         C=R3                Check if LOOP
2623            *
```

```
2624 F4E4D 97A          ?C=0   W            LOOP?
2625 F4E50 80           GOYES  NEWF75       Yes...don't rewind!
2626 F4E52 8CA1         GOLONG ENDTAP       Carry = result
           7F
2627               *-
2628               *-
2629 F4E58 96F  NEWF75  ?D#0   B            Is this "LOOP"?
2630 F4E5B C0           GOYES  Utlend       No...clean up
2631 F4E5D 3100         LC(2)  =mENDM       Yes...set ETO
2632 F4E61 8C00         GOLONG =PUTC+
           00
2633               *-
2634               *-
2635 F4E67 8C00 =Utlend GOLONG =UTLEND      Unt, Unl, END
           00
2636               *-
2637               *-
2638               *
2639               * 8: Subroutine to write the new directory entry to the medium
2640               *
2641 F4E6D      NEWF80
2642               *
2643               * First check if found the old file (If found and writing
2644               * somewhere else, purge this first)
2645               *
2646               * IF (PTRL#0 AND PTRL#PTRD) THEN PTRL_file_type <== 0
2647               *
2648               * First check PTRL#0
2649               *
2650 F4E6D AF4            A=B    W            Get PTRL into A[11:8]...
2651 F4E70 8E00           GOSUBL =ASRC8       ...move to A[3:0]...
           00
2652 F4E76 23           P=     3
2653 F4E78 91C          ?A#0   WP           ...and check if non-zero
2654 F4E7B 60           GOYES  NEWF8!       Non-zero...check PTRL#PTRD
2655 F4E7D 6E60 NEWF8.  GOTO   NEWF82       Zero...continue
2656               *-
2657               *-
2658               *
2659               * Now check PTRL#PTRD...Use A to fetch PTRD
2660               *
2661 F4E81 AF9  NEWF8!  C=B    W            Get PTRD into C[15:12]...
2662 F4E84 731C         GOSUB  Csrc12       ...shift into C[3:0]...
2663 F4E88 912          ?A=C   WP           ...and check for equality
2664 F4E8B 2F           GOYES  NEWF8.       EQUAL...skip purge
2665               *
2666               * Need to purge the file here (PTRL is in A[3:0])
2667               *
2668               * If this purge were to be done when it is FOUND, there will
2669               * be less medium wear, but the file would be purged even if an
2670               * error occurs while trying to create the new file
2671               *
2672 F4E8D 814              ASRC           Shift PTRL - get record # in A[X]
2673               *
2674               * Now A[X] is the record #, A[S] is the directory entry #
```

```
2675                *
2676 F4E90 A80        A=0    P           (P is still 3 from above stmts)
2677 F4E93 73BB       GOSUB  Seeka       Go to that record
2678 F4E97 400        RTNC               Error
2679 F4E9A 7FDB       GOSUB  Mtyl        Send DDL to the drive
2680 F4E9E 400        RTNC
2681                *
2682                * Read the record into buffer zero of the drive
2683                *
2684 F4EA1 8E00       GOSUBL =DdlPwr     Send partial write mode, MTYL
           00
2685 F4EA7 400        RTNC               Error
2686                *
2687                * Set the drive mode back to WRITE mode (NOT partial write)
2688                *
2689 F4EAA 778B       GOSUB  DdlWrt      Write mode (Sets Byte pointer=0)
2690 F4EAE 400        RTNC
2691                *
2692                * Now buffer 0 contains the record...modify the file type
2693                * at PTRL (set to zero) and write the record out to the medium
2694                *
2695 F4EB1 20         P=     =SetBP      Set byte pointer
2696 F4EB3 708B       GOSUB  Ddl
2697 F4EB7 400        RTNC
2698 F4EBA 810        ASLC               Move entry # to A[0]
2699 F4EBD F0         ASL    A           Shift into the B field (*16)
2700 F4EBF C4         A=A+A  A           Double it (*32)
2701 F4EC1 31A0       LC(2)  10          Byte # within entry of file type
2702 F4EC5 A62        C=C+A  B           Now C[B] points to the file type
2703 F4EC8 75AB       GOSUB  Putd        Send it!
2704 F4ECC 400        RTNC
2705                *       P=     =Write0     (Write0 is 0, P is already 0)
2706 F4ECF 746B       GOSUB  Ddl         Set WRITE mode
2707 F4ED3 400        RTNC
2708 F4ED6 D2         C=0    A           Clear C[B]
2709 F4ED8 759B       GOSUB  Putd        Send first byte of type (PURGED)
2710 F4EDC 400        RTNC
2711 F4EDF 3300       LC(4)  =mENDf      Send last byte as an END frame
           00
2712 F4EE5 7F6B       GOSUB  Putc
2713 F4EE9 400        RTNC
2714 F4EEC          NEWF82
2715                *
2716                * Now ready to write the new entry (Create it in SCRTCH first)
2717                *
2718 F4EEC 7C3B       GOSUB  D1=SCR      Name...
2719 F4EF0 110        A=R0               (First 8 chars)
2720 F4EF3 1517       DAT1=A W           (Write first 8 chars)
2721 F4EF7 17F        D1=D1+ 16
2722                *
2723                * At this point, save the contents of R1 @(=SCRTCH)+#10, B[W]
2724                * @(=SCRTCH)+#20, D0 @(=SCRTCH)+#30, and D[A,15:13] @(=SCRTCH)
2725                * +#35 so that I can call YMDHMS, which uses D0,D1,A-D,R0,R1
2726                *
2727 F4EFA 119        C=R1
```

```
2728 F4EFD 1557          DAT1=C W            Save R1 @ (=SCRTCH)+#10
2729 F4F01 17F           D1=D1+ 16
2730 F4F04 AF9           C=B    W
2731 F4F07 1557          DAT1=C W            Save B @ (=SCRTCH)+#20
2732 F4F0B 17F           D1=D1+ 16
2733 F4F0E 136           CDOEX
2734 F4F11 145           DAT1=C A            Save DO @ (=SCRTCH)+#30
2735 F4F14 174           D1=D1+ 5
2736 F4F17 AFB           C=D    W
2737 F4F1A 708B          GOSUB  Cslc3
2738 F4F1E 15D7          DAT1=C 8            Save D[A,15:13] @ (=SCRTCH)+#35
2739              *
2740              * Now I am ready to call YMDHMS
2741              *
2742 F4F22 748B          GOSUB  Ymdhms       Returns with info in C[11:0]
2743              *
2744              * Registers A,B,D,DO,D1,RO,R1 are NOT defined now!
2745              *
2746              * Restore registers and write out the info
2747              *
2748 F4F26 AF7           D=C    W            Save time in D[W] for now
2749 F4F29 11C           C=R4
2750 F4F2C 7B6B          GOSUB  Csrc12       Get last 2 chars in C[3:0]
2751 F4F30 1F00          D1=(5) (=SCRTCH)+16 Point to filename
          000
2752 F4F37 1537          A=DAT1 W            Read in R1 from =SCRTCH+#10
2753 F4F3B 101           R1=A                Restore it
2754 F4F3E 15D3          DAT1=C 4            Write out last two chars of name
2755 F4F42 173           D1=D1+ 4
2756 F4F45 7B5B          GOSUB  Asrc4
2757 F4F49 AF6           C=A    W
2758 F4F4C 7BB1          GOSUB  PT2BYT       Output file type
2759 F4F50 AF2           C=0    W
2760 F4F53 15D7          DAT1=C 8            Clear out start address field
2761 F4F57 177           D1=D1+ 8            Move to B[W] save area
2762              *
2763              * Set start address <== PTRF
2764              *
2765 F4F5A 1577          C=DAT1 W            PTRF is in C[7:4]...
2766 F4F5E AF5           B=C    W            ...(restore B[W])...
2767 F4F61 8E00          GOSUBL =CSRC2       ...shift into C[5:2]...
          00
2768 F4F67 1C3           D1=D1- 4            ...position to START field...
2769 F4F6A 7D91          GOSUB  PT2BYT       ...Put 2 bytes, D1=D1+ 4
2770              *
2771              * D1 now points @ (=SCRTCH)+ #20 (LENGTH field)
2772              *
2773 F4F6E AF2           C=0    W
2774 F4F71 15D3          DAT1=C 4            Clear first 2 bytes of LENGTH
2775 F4F75 173           D1=D1+ 4            Skip to second half!
2776              *
2777              * Set length field <== (NSize + 255) DIV 256
2778              *
2779 F4F78 119           C=R1                NSize is in C[5:0]!
2780 F4F7B 96A           ?C=0   B            Is this an even # of records?
```

```
2781 F4F7E 61          GOYES  NEWF8,        Yes...continue
2782 F4F80 B26         C=C+1  XS            No...add 1 to it!
2783 F4F83 550         GONC   NEWF83        If carry, propagate into C[M]
2784 F4F86 B56         C=C+1  M
2785 F4F89 97D NEWF83  ?B#0   W             Loop?
2786 F4F8C 80          GOYES  NEWF8,        No...continue
2787 F4F8E AE2         C=0    B             Yes...
2788 F4F91 109         R1=C                 ...set length=# recs * 256
2789 F4F94     NEWF8,
2790           *
2791           * Now C[5:2] is length in records
2792           *
2793 F4F94 7371        GOSUB  PT2BYT        Put 2 bytes, increment D1 by 4
2794           *
2795           * D1 is now @ (=SCRTCH)+ #28 (time of creation field)
2796           *
2797 F4F98 177         D1=D1+ 8             Skip to saved DO
2798 F4F9B 147         C=DAT1 A
2799 F4F9E 147         C=DAT1 A             Read in DO...
2800 F4FA1 134         DO=C                 ...restore it
2801 F4FA4 174         D1=D1+ 5
2802 F4FA7 15F7        C=DAT1 8             Read in D stuff...
2803 F4FAB 70EA        GOSUB  Csrc3         ...rotate to correct place...
2804 F4FAF AFF         CDEX   W             ...and put in D[W], fetch time
2805 F4FB2 1CC NEWF84  D1=D1- 13            Back up to start of time field
2806           *
2807           * Output it in the proper order!
2808           *
2809 F4FB5 2A          P=     16-6          Increment P until carry...6 times
2810 F4FB7 7ADA        GOSUB  Cslc6         Move to C[B],C[15:6]
2811 F4FBB 14D NEWF85  DAT1=C B             Write this byte...
2812 F4FBE 171         D1=D1+ 2             ...move to next byte...
2813 F4FC1 8E00        GOSUBL =CSLC2        ...shift in next byte...
           00
2814 F4FC7 0C          P=P+1                ...increment count...
2815 F4FC9 51F         GONC   NEWF85        ...if no carry, continue!
2816           *
2817           * Now output volume number, END flag
2818           *
2819 F4FCC 22          P=     2
2820 F4FCE 3310        LCHEX  8001          Volume 1, END
           08
2821 F4FD4 7331        GOSUB  PT2BYT        Put 2 bytes from C[5:2]
2822           *
2823           * D1 is now at the implementation bytes
2824           *
2825 F4FD8 11A         C=R2                 Get NImpl from R2[7:0]
2826 F4FDB 15D7        DAT1=C 8             Write them out!
2827 F4FDF 97D         ?B#0   W             LOOP or non-MS device?
2828 F4FE2 01          GOYES  NEWF87        No...continue
2829 F4FE4 96B         ?D=0   B             LOOP?
2830 F4FE7 66          GOYES  NEWF97        Yes...skip addressing!
2831           *
2832           * Non-mass storage...address me as talker, device as Listener
2833           *
```

```
2834 F4FE9 709A          GOSUB  Mtyl          Controller...address me as talker
2835 F4FED 5F5           GONC   NEWF97        Go if no error
2836 F4FF0 02            RTNSC                Return with error
2837              *-
2838              *-
2839              *
2840              * Now entry is created in SCRTCH...write it to the medium
2841              *
2842 F4FF2 1F00  NEWF87  D1=(5) (=SCRTCH)+36
          000
2843 F4FF9 75F0          GOSUB  GT2BYO        Read 2 bytes (size in records)
2844 F4FFD 10A           R2=C                 Save size of file in R2[A]
2845              *
2846 F5000 AF9           C=B    W             Copy PTRD into C[15:12]...
2847 F5003 D2            C=0    A             ...clear nibbles "above" PTRD...
2848 F5005 759A          GOSUB  Csrc13        ...shift into C[2:0], C[S]...
2849 F5009 AFA   =PUTDR# A=C    W             ...save all in A[W]...
2850 F500C 7A3A          GOSUB  Seeka         ...goto the correct record
2851 F5010 400           RTNC
2852 F5013 766A          GOSUB  Mtyl          Make me talker, drive as listener
2853 F5017 400           RTNC
2854 F501A       NEWF90
2855 F501A 8E00          GOSUBL =DdlPwr       Partial write mode, check status
          00
2856 F5020 400           RTNC
2857              *
2858              * Set back to write mode before sending data to drive
2859              *
2860 F5023 7E0A          GOSUB  DdlWrt        Write mode
2861 F5027 400           RTNC
2862              *
2863              * Set byte pointer to current position
2864              *
2865 F502A 20            P=     =SetBP
2866 F502C 770A          GOSUB  Ddl           Set byte pointer
2867 F5030 400           RTNC
2868 F5033 810           ASLC                 Get entry number back to A[0]
2869 F5036 AE6           C=A    B
2870 F5039 F2            CSL    A             C[B] is now entry number * 16...
2871 F503B C6            C=C+C  A             ...* 32...
2872 F503D 703A          GOSUB  Putd          ...Send to the drive
2873 F5041 400           RTNC
2874              *
2875              * Set back to WRITE mode
2876              *
2877 F5044       =PUTDIR
2878              *
2879              * Entry to write a directory entry from SCRTCH
2880              *
2881 F5044 20            P=     =WriteO       Write mode (resume)
2882 F5046 7DE9 =PUTDR"  GOSUB  Ddl
2883 F504A 400           RTNC
2884              *
2885              * Now send the entry to the drive
2886              *
```

```
2887 F504D       NEWF97
2888 F504D 7BD9          GOSUB  D1=SCR        Point to the entry...
2889 F5051 D2            C=0    A
2890 F5053 20            P=     0             P could be non-zero from jump in
2891 F5055 31F1          LC(2)  31            Send all but the last byte.
2892 F5059 DA            A=C    A
2893 F505B 6160          GOTO   NEWF.3        (WRITIT,mENDf, check drive status)
2894             *-
2895             *-
2896 F505F 119   NEWF.0   C=R1                Get NData into C[14:10]
2897 F5062 25            P=     5
2898 F5064 91E           ?C#0   WP            Is the file size zero?
2899 F5067 40            GOYES  NEWF.1        No...seek to the data area?
2900 F5069 03    NEWF.c  RTNCC                Yes...don't seek to the data area
2901             *-
2902             *-
2903 F506B 762A  NEWF.1  GOSUB  Csrc10         Shift to C[4:0]
2904 F506F 8AE           ?C#0   A             Is NData zero? (no copy)
2905 F5072 F0            GOYES  NEWF.2        No...continue on
2906             *
2907             * NData is zero...no data address to copy (check if CREATE)
2908             *
2909 F5074 25            P=     15-10         Point at R1[S]
2910 F5076 90A           ?C=0   P             Is this a COPY?
2911 F5079 0F            GOYES  NEWF.c        Yes...don't seek to the data area
2912 F507B B06           C=C+1  P             Is this a non-mass storage device?
2913 F507E 4AE   NEWF.C  GOC    NEWF.c        Yes...don't seek!
2914 F5081 135   NEWF.2  D1=C                 Set D1 <== start of data
2915 F5084 979           ?B=0   W             LOOP?
2916 F5087 02            GOYES  NEWF98        Yes...skip SEEK
2917 F5089 AF9           C=B    W
2918 F508C 7CF9          GOSUB  Csrc4         Get PTRF into C[3:0]...
2919 F5090 DA            A=C    A             ...Copy to A[3:0]...
2920 F5092 74B9          GOSUB  Seeka         ...and SEEK to that record
2921 F5096 400           RTNC
2922 F5099 70E9          GOSUB  Mtyl          I must be talker to do DDLs
2923 F509D 400           RTNC
2924 F50A0 7199          GOSUB  DdlWrt        Write mode...
2925 F50A4 400           RTNC
2926 F50A7 111   NEWF98  A=R1                Copy NSize to A[A]...
2927 F50AA CC            A=A-1  A             ...leave 1 byte to END...
2928 F50AC 948           ?A=0   S             Called by COPY?
2929 F50AF E0            GOYES  NEWF.3        If so, copy it
2930 F50B1 B44           A=A+1  S             LOOP?
2931 F50B4 480           GOC    NEWF.3        Yes...copy it
2932 F50B7 8C00          GOLONG =INITFL       Initialize file if CCode#0
          00
2933             *-
2934             *-
2935 F50BD 7D99  NEWF.3  GOSUB  Writit        Send (NSize) bytes to the device
2936 F50C1 400           RTNC
2937             *
2938             * Because the ENDf message is a SEND message, make sure I am
2939             * active talker first (otherwise will get Invalid Mode error)
2940             *
```

```
2941 F50C4 8E00 NEWF..  GOSUBL =GETST       Get status...(sets P=0)
           00
2942 F50CA 400          RTNC
2943 F50CD 0B           CSTEX
2944 F50CF 860          ?ST=0  =sTALKA      Talker active?
2945 F50D2 20           GOYES  NEWF.,       (Set carry if not)
2946 F50D4 0B   NEWF.,  CSTEX
2947 F50D6 4DE          GOC    NEWF..       Not talker active...wait!
2948 F50D9 3300         LC(4)  =mENDf       End frame
           00
2949 F50DF 14F          C=DAT1 B            Read value of last data byte
2950 F50E2 7279         GOSUB  Putc         Send the last frame as an END
2951 F50E6 400          RTNC
2952 F50E9 979          ?B=0   W            LOOP?
2953 F50EC 29           GOYES  NEWF.C       Yes...return, carry clear
2954 F50EE 6F59         GOTO   Tstat        Check drive status! (carry=status)
2955            ****************************************************************
2956            ****************************************************************
2957            **
2958            ** Name:       GETBYT - Read bytes from RAM (most sig. first)
2959            **
2960            ** Category:   LOCAL
2961            **
2962            ** Purpose:
2963            **      Read "P" bytes from RAM into C from D1 (Bytes are high
2964            **      bytes first)
2965            **
2966            ** Entry:
2967            **      P= # of bytes to read - 1
2968            **      D1 points to first byte
2969            **
2970            ** Exit:
2971            **      P=0
2972            **      Carry clear
2973            **      C contains (P+1) bytes of data
2974            **      D1 points to the next byte (first one NOT used)
2975            **
2976            ** Calls:      None
2977            **
2978            ** Uses.......
2979            **  Inclusive: C[W],D1,P  (Unused nibbles of C shifted left)
2980            **
2981            ** Stk lvls:   0
2982            **
2983            ** History:
2984            **
2985            **    Date     Programmer          Modification
2986            **  --------   ----------   ---------------------------------
2987            **  11/19/82      NZ        Added documentation
2988            **
2989            ****************************************************************
2990            ****************************************************************
2991 F50F2 D2   =GT2BYO C=0    A            Clear C[A] first
2992 F50F4 21   =GT2BYT P=     1            Read 2 bytes
2993 F50F6 BF2  =GETBYT CSL    W            Preshift C over one byte
```

```
2994 F50F9 BF2          CSL    W
2995 F50FC 14F          C=DAT1 B
2996 F50FF 171          D1=D1+ 2
2997 F5102 0D           P=P-1                Is this the end?
2998 F5104 51F          GONC   GETBYT        No...get another
2999 F5107 20           P=     0             Set P=0
3000 F5109 03           RTNCC
3001            ****************************************************************
3002            ****************************************************************
3003            **
3004            ** Name:       PT2BYT - Wtite 2 bytes, high byte first, to RAM
3005            **
3006            ** Category:   LOCAL
3007            **
3008            ** Purpose:
3009            **      Output 2 bytes at D1 from C[5:2] (C[5:4] first,then
3010            **      C[3:2])
3011            **
3012            ** Entry:
3013            **      C[5:2] contains the two bytes
3014            **      D1 points to destination RAM
3015            **
3016            ** Exit:
3017            **      D1 points to first byte following the written data
3018            **      Carry clear
3019            **
3020            ** Calls:      CSRC4
3021            **
3022            ** Uses.......
3023            **  Exclusive:   D1
3024            **  Inclusive: C[W],D1 (C[W] is shifted right circular 4 nibs)
3025            **
3026            ** Stk lvls:   1 (CSRC4)
3027            **
3028            ** History:
3029            **
3030            **    Date      Programmer              Modification
3031            **  --------    ----------    ---------------------------------
3032            **  11/19/82      NZ          Added documentation
3033            **
3034            ****************************************************************
3035            ****************************************************************
3036 F510B 15D3 =PT2BYT DAT1=C 4              Write the low byte first...
3037 F510F 7979         GOSUB  Csrc4          ...get the high byte into C[B]...
3038 F5113 14D          DAT1=C B              ...write the high byte
3039 F5116 173          D1=D1+ 4              Increment D1 past data...
3040 F5119 03           RTNCC                 ...and return with carry clear
3041 F511B              END
```

```
 ASLC3   Ext                  -  1539
 ASLC4   Ext                  -   406   421   645
 ASLC9   Ext                  -  1441
 ASRC10  Ext                  -  1022
 ASRC3   Ext                  -  1536
 ASRC4   Ext                  -  1695
 ASRC5   Ext                  -   964
 ASRC8   Ext                  -  2651
 ASRC9   Ext                  -  1522
 Asrc4   Abs 1002148 #F4AA4   -  1695   424  1460  2377  2756
 BLANKC  Ext                  -   466
=CHKBIT  Abs 1000206 #F43OE   -   212   973  1004  1017  1107  2057
=CHKMAS  Abs 1000177 #F42F1   -   169
 CHKMAe  Abs 1000197 #F4305   -   177   173  1112
 CHKSEC  Ext                  -  2211
 CHKSIZ  Abs 1002746 #F4CFA   -  2376  2367  2536
 CHKSIz  Abs 1002777 #F4D19   -  2396  2388
=CLEARN  Abs 1000216 #F4318   -   257   654
=CLLOOP  Abs 1000221 #F431D   -   259   652
 CSLC10  Ext                  -   945
 CSLC2   Ext                  -  2813
 CSLC3   Ext                  -  1692
 CSLC8   Ext                  -  2330
 CSRC2   Ext                  -  2767
 CSRC3   Ext                  -  1683
 CSRC8   Ext                  -  1531  2144
 CSRC9   Ext                  -  2532
 ChkEOT  Abs 1000681 #F44E9   -   590   351   570
 ChkEOt  Abs 1000690 #F44F2   -   593   592
 Cslc12  Abs 1002124 #F4A8C   -  1681  2588
 Cslc3   Abs 1002142 #F4A9E   -  1692  2232  2737
 Cslc4   Abs 1002139 #F4A9B   -  1690   318   506  1540  2273  2423
 Cslc5   Abs 1002136 #F4A98   -  1688   627   901   938   950  1029  1213  2299
                              2465
 Cslc6   Abs 1002133 #F4A95   -  1687  2810
 Csrc10  Abs 1002133 #F4A95   -  1686   928   943   988  2903
 Csrc12  Abs 1002139 #F4A9B   -  1689  2182  2585  2662  2750
 Csrc13  Abs 1002142 #F4A9E   -  1691  2848
 Csrc3   Abs 1002127 #F4A8F   -  1683  2235  2803
 Csrc4   Abs 1002124 #F4A8C   -  1682  2278  2289  2432  2918  3037
 Csrc5   Abs 1002121 #F4A89   -  1680   639  1007  1027  1208  2307  2456  2556
 D1=AVE  Ext                  -   907
=D1=SCR  Abs 1002028 #F4A2C   -  1634   548   609  1121  1354  1414  1504  1623
                              1664  2243  2718  2888
 D1@AVS  Ext                  -   997
 DDL     Ext                  -  1639
 DDT     Ext                  -  1654
 Ddl     Abs 1002039 #F4A37   -  1639   110   430   711   714   840  1336  2696
                              2706  2866  2882
 DdlPwr  Ext                  -  2684  2855
 DdlWrt  Abs 1002037 #F4A35   -  1638   447   830  1019  2689  2860  2924
 Ddt     Abs 1002066 #F4A52   -  1654   335   558   767   770   779  1643
=DdtRd   Abs 1002045 #F4A3D   -  1642   764   985  1329  1407  2124
=ENDTAP  Abs 1000814 #F456E   -   706  2626
 F->SC!  Abs 1002014 #F4A1E   -  1627  1630
```

```
=F->SCR  Abs 1002002 #F4A12 -  1623   660  2599
=FINDF+  Abs 1001275 #F473B -  1098
 FINDF0  Abs 1001425 #F47D1 -  1170  1219
 FINDF1  Abs 1001477 #F4805 -  1197  1183  1189
 FINDF2  Abs 1001510 #F4826 -  1218  1203
 FINDF3  Abs 1001546 #F484A -  1243  1195
 FINDF4  Abs 1001549 #F484D -  1247  1142
=FINDFL  Abs 1001268 #F4734 -  1094
 FINDFe  Abs 1001576 #F4868 -  1261  1249  1255
 FINDFl  Abs 1001395 #F47B3 -  1153  1105
 FINDFn  Abs 1001519 #F482F -  1224  1180  1212
=FINDFx  Abs 1001415 #F47C7 -  1164  1108
 FINDf+  Abs 1001278 #F473E -  1099  1095
 FINDfn  Abs 1001385 #F47A9 -  1145  1135  1140
 FIND12  Abs 1001327 #F476F -  1121  1158
 FIND14  Abs 1001378 #F47A2 -  1141  1133  1145
 FINDle  Abs 1001391 #F47AF -  1149  1154
 FIXSPC  Ext                -   597
 FORM10  Abs 1000257 #F4341 -   312   302
 FORM20  Abs 1000280 #F4358 -   328   326
 FORM30  Abs 1000332 #F438C -   365   350
 FORM50  Abs 1000353 #F43A1 -   380   358
 FORM60  Abs 1000367 #F43AF -   391   381
 FORM65  Abs 1000390 #F43C6 -   412   403
 FORM70  Abs 1000394 #F43CA -   416   408
=FORMAT  Abs 1000230 #F4326 -   301   307
 Format  Ext                -   429
 GDIRS1  Abs 1001746 #F4912 -  1424
 GDIRS3  Abs 1001780 #F4934 -  1441  1429
 GDIRS4  Abs 1001832 #F4968 -  1478  1470
 GDIRS7  Abs 1001883 #F499B -  1517  1486  1500
 GDIRS8  Abs 1001890 #F49A2 -  1520  1475
 GDIRS9  Abs 1001892 #F49A4 -  1521  1516
 GDIRSE  Abs 1001765 #F4925 -  1430  1453
 GDIRSM  Abs 1001965 #F49ED -  1564  1499  1509
=GDIRST  Abs 1001688 #F48D8 -  1404  1322  2069
 GDIRSe  Abs 1001775 #F492F -  1436  1423
 GDIRSm  Abs 1001980 #F49FC -  1574  1577
 GDIRsE  Abs 1001767 #F4927 -  1431  1438
 GDIRsM  Abs 1001993 #F4A09 -  1584  1575
 GETALR  Ext                -  1416  1427  1455
=GETBYT  Abs 1003766 #F50F6 -  2993  2998
 GETD    Ext                -  1237
=GETDIR  Abs 1001653 #F48B5 -  1349  1218  1332
=GETDR!  Abs 1001580 #F486C -  1322  1165
=GETDR"  Abs 1001587 #F4873 -  1324
=GETDR#  Abs 1001589 #F4875 -  1325
=GETDR+  Abs 1001614 #F488E -  1333
 GETDev  Ext                -   968  1002
 GETST   Ext                -  2941
 GETZER  Ext                -  1234
=GT2BYO  Abs 1003762 #F50F2 -  2991  2204  2843
=GT2BYT  Abs 1003764 #F50F4 -  2992  2231  2257  2272  2363  2411  2414
 GTYPE   Ext                -   169
 Getd    Abs 1001534 #F483E -  1237    56   349   373   565
```

```
 GetMbx  Ext                  -  2309  2619
 Getzer  Abs 1001528 #F4838   -  1234  1248  1254  1485  1564
 INIT05  Abs 1000493 #F442D   -   467   465
 INIT10  Abs 1000697 #F44F9   -   604   566
 INIT20  Abs 1000706 #F4502   -   609   587
 INITFL  Ext                  -  2932
=INITIL  Abs 1000438 #F43F6   -   442
 ImpByt  Ext                  -   557
 LSTEN1  Abs 1002197 #F4AD5   -  1750  1741  1742  1749
=LSTENT  Abs 1002185 #F4AC9   -  1745
 LoopOK  Ext                  -  1660
 MOVEF,  Abs 1001114 #F469A   -   982   974
 MOVEF1  Abs 1000978 #F4612   -   907  1035
 MOVEF2  Abs 1001031 #F4647   -   934   930
 MOVEF3  Abs 1001040 #F4650   -   938   935
 MOVEF4  Abs 1001127 #F46A7   -   986   976
 MOVEF5  Abs 1001207 #F46F7   -  1015  1005
 MOVEF6  Abs 1001228 #F470C   -  1021  1003  1018
=MOVEFL  Abs 1000966 #F4606   -   898
 MOVEd1  Abs 1001130 #F46AA   -   987   969
 MTYL    Ext                  -  1674
 MaxRec  Ext                  -   334
 Mtyl    Abs 1002109 #F4A7D   -  1674   107   427   445   613   708   828  1015
                              1333  2594  2679  2834  2852  2922
 NEWF++  Abs 1002225 #F4AF1   -  2059  2053
 NEWF.,  Abs 1003732 #F50D4   -  2946  2945
 NEWF..  Abs 1003716 #F50C4   -  2941  2947
 NEWF.0  Abs 1003615 #F505F   -  2896  2605
 NEWF.1  Abs 1003627 #F506B   -  2903  2899
 NEWF.2  Abs 1003649 #F5081   -  2914  2905
 NEWF.3  Abs 1003709 #F50BD   -  2935  2893  2929  2931
 NEWF.C  Abs 1003646 #F507E   -  2913  2953
 NEWF.c  Abs 1003625 #F5069   -  2900  2911  2913
 NEWF01  Abs 1002250 #F4B0A   -  2069  2064  2115
 NEWF03  Abs 1002271 #F4B1F   -  2113  2112
 NEWF05  Abs 1002281 #F4B29   -  2121  2104
 NEWF1.  Abs 1002847 #F4D5F   -  2449  2467
 NEWF10  Abs 1002298 #F4B3A   -  2129  2449
 NEWF11  Abs 1002331 #F4B5B   -  2138  2150
 NEWF12  Abs 1002335 #F4B5F   -  2141  2137
 NEWF13  Abs 1002365 #F4B7D   -  2159  2142  2146
 NEWF14  Abs 1002396 #F4B9C   -  2176  2175
 NEWF15  Abs 1002683 #F4CBB   -  2325  2263
 NEWF1a  Abs 1002417 #F4BB1   -  2191  2184
 NEWF1b  Abs 1002552 #F4C38   -  2271  2262
 NEWF1c  Abs 1002470 #F4BE6   -  2225  2207  2212
 NEWF1d  Abs 1002466 #F4BE2   -  2218  2194
 NEWF2.  Abs 1002357 #F4B75   -  2153  2165
 NEWF20  Abs 1002701 #F4CCD   -  2338  2153
 NEWF25  Abs 1002703 #F4CCF   -  2339  2340
 NEWF3.  Abs 1002361 #F4B79   -  2156  2177  2185
 NEWF30  Abs 1002713 #F4CD9   -  2349  2156
 NEWF34  Abs 1002784 #F4D20   -  2401  2358  2368
 NEWF4.  Abs 1002709 #F4CD5   -  2341  2350
 NEWF40  Abs 1002838 #F4D56   -  2441  2341  2373
```

```
 NEWF45  Abs 1002851 #F4D63 -  2455  2448
 NEWF48  Abs 1002877 #F4D7D -  2470  2464
 NEWF50  Abs 1002882 #F4D82 -  2475  2138
 NEWF55  Abs 1002887 #F4D87 -  2484  2066
 NEWF60  Abs 1002898 #F4D92 -  2492  2480
 NEWF61  Abs 1002921 #F4DA9 -  2513  2537
 NEWF62  Abs 1002929 #F4DB1 -  2521  2500  2506
 NEWF66  Abs 1002982 #F4DE6 -  2571  2560
 NEWF67  Abs 1002995 #F4DF3 -  2584  2551  2570
 NEWF70  Abs 1003046 #F4E26 -  2605  2319  2486  2578  2580
 NEWF75  Abs 1003096 #F4E58 -  2629  2625
 NEWF8!  Abs 1003137 #F4E81 -  2661  2654
 NEWF8,  Abs 1003412 #F4F94 -  2789  2781  2786
 NEWF8.  Abs 1003133 #F4E7D -  2655  2664
 NEWF80  Abs 1003117 #F4E6D -  2641  2484  2541
 NEWF82  Abs 1003244 #F4EEC -  2714  2655
 NEWF83  Abs 1003401 #F4F89 -  2785  2783
 NEWF84  Abs 1003442 #F4FB2 -  2805  2317
 NEWF85  Abs 1003451 #F4FBB -  2811  2815
 NEWF87  Abs 1003506 #F4FF2 -  2842  2828
 NEWF90  Abs 1003546 #F501A -  2854  2600
 NEWF97  Abs 1003597 #F504D -  2887  2830  2835
 NEWF98  Abs 1003687 #F50A7 -  2926  2916
=NEWFI+  Abs 1002206 #F4ADE -  2045
=NEWFIL  Abs 1002234 #F4AFA -  2062  2058
 NEWFeT  Abs 1002917 #F4DA5 -  2509  2515
=NXTEN+  Abs 1002161 #F4AB1 -  1733  1201  2446  2550
=NXTENT  Abs 1002163 #F4AB3 -  1734  2431  2586
 PRMSGA  Ext                -   479
=PT2BYT  Abs 1003787 #F510B -  3036  2291  2758  2769  2793  2821
 PUGFIB  Ext                -  2618
 PURFIB  Ext                -  2239
 PUTALR  Ext                -   647
 PUTC    Ext                -  1657
 PUTC+   Ext                -  2632
 PUTD    Ext                -  1668
=PUTDIR  Abs 1003588 #F5044 -  2877   661
=PUTDR"  Abs 1003590 #F5046 -  2882
=PUTDR#  Abs 1003529 #F5009 -  2849
 PUTDX   Ext                -  1671
 PUTE    Ext                -    54   347   563
 Putc    Abs 1002072 #F4A58 -  1657  2712  2950
 Putd    Abs 1002097 #F4A71 -  1668   115   118   489   513  1342  2703  2709
                            -  2872
 Putdx   Abs 1002103 #F4A77 -  1671   452   486   500   537   542
 READI3  Ext                -   604
=READR#  Abs 1000852 #F4594 -   761
 READSU  Ext                -  1665
 Read    Ext                -  1642
 Read1   Ext                -   769
 Readsc  Abs 1002087 #F4A67 -  1664  1352  1412  1479  2131
 Readsu  Abs 1002091 #F4A6B -  1665   776   992  1122
 Rewind  Ext                -   713
 Rtncc   Abs 1000195 #F4303 -   174
 SCRTCH  Ext                -  1128  1484  1634  2133  2198  2230  2252  2283
```

```
                                    2290  2310  2362  2410  2751  2842
=SEEKA   Abs 1000135 #F42C7 -        107   443   762   826  1327  1405  1648
=SEEKB   Abs 1000142 #F42CE -        109
 SENDIT  Ext                 -        260
 START   Ext                 -       1677
 Seek    Ext                 -        109
 Seeka   Abs 1002058 #F4A4A -        1648   983  1013  2122  2592  2677  2850  2920
 SetBP   Ext                 -       1335  2695  2865
 Start   Abs 1002115 #F4A83 -        1677   958  1000  1102  2045
=TSTAT   Abs 1000083 #F4293 -          50   123   301   432   706   761   825   842
                                    1651
 TSTAT1  Abs 1000109 #F42AD -          56
 TSTAT2  Abs 1000127 #F42BF -          64    60
=TSTATA  Abs 1000090 #F429A -          52    66   784  1349  1645
 Tstat   Abs 1002062 #F4A4E -        1651  2954
 Tstata  Abs 1002054 #F4A46 -        1645
 UTLEND  Ext                 -        716  2635
=Utlend  Abs 1003111 #F4E67 -        2635  2630
=WRITE#  Abs 1000916 #F45D4 -         825
 WRITIT  Ext                 -       1661
 Write   Ext                 -       1638
 Write0  Ext                 -       2881
 Writit  Abs 1002078 #F4A5E -        1660   621   837  1033  2935
 XchgT   Ext                 -        766   778
 YMDHMS  Ext                 -       1698
 YTML    Ext                 -       1240
 Ymdhms  Abs 1002154 #F4AAA -        1698   633  2301  2742
 Ytml    Abs 1001540 #F4844 -        1240    50   549   975  1118  1344
 bFIB    Ext                 -       2233
 eDIRFL  Ext                 -       2508
 eDSPEC  Ext                 -       1149
 eDTYPE  Ext                 -        178
 eEFILE  Ext                 -       2192
 eEOTAP  Ext                 -       2514
 eNEWTA  Ext                 -        306  2111
 eNFILE  Ext                 -       1229
 eNOLIF  Ext                 -       1437
 eNORAM  Ext                 -        921
 ePIL    Ext                 -        179
 eRANGE  Ext                 -        412  1265
 eTAPE   Ext                 -        303  1230  1432  2108  2218  2509
 eTSIZE  Ext                 -       1430
 efPROT  Ext                 -       2217
=fPROT   Abs 1002461 #F4BDD -        2216
 fTYPF#  Ext                 -       2206
 hCPY5s  Ext                 -        991  1157
 i/OFND  Ext                 -       2234
 mENDM   Ext                 -       2631
 mENDf   Ext                 -       2711  2948
 mSDA    Ext                 -        346   561   772  1120  1351  1410  1478  2130
 mSST    Ext                 -         53
 pEOT    Ext                 -        591
 sLoop?  Ext                 -       1094  1098  1113  1153  1164
 sOVERW  Ext                 -       2193
 sTALKA  Ext                 -       2944
```

Input Parameters

Source file name is NZ&CAS::MS

Listing file name is NZ/CAS:TI:ML::-1

Object file name is NZ%CAS:TI:MS::-1

```
                                  111111
                        0123456789012345
```

Initial flag settings are

Errors

None

Saturn Assembler News

```
     1            *
     2            *       N   N  ZZZZZ   &     H   H  N   N  D DD
     3            *       N   N      Z  & &   H   H  N   N  D   D
     4            *       NN  N     Z   & &   H   H  NN  N  D   D
     5            *       N N N    Z     &    HHHHH  N N N  D   D
     6            *       N  NN   Z     & & &  H   H  N  NN  D   D
     7            *       N   N  Z      &  &   H   H  N   N  D   D
     8            *       N   N  ZZZZZ   && &  H   H  N   N  DDDD
     9            *
    10            *
    11                    TITLE  POLL HANDLERS <840106.0805>
    12 F511B              ABS    #F511B         TI%HP6 address (fixed)
    13                    RDSYMB  TI%EQU
```

```
14                      STITLE DATA FILE HANDLERS
15                *
16                ****************************************************************
17                ****************************************************************
18                **
19                ** Name:       hVER$ - Handler for the VER$ poll
20                **
21                ** Category:   POLL
22                **
23                ** Purpose:
24                **      Add HPIL info to the VER$ string
25                **
26                ** Entry:
27                **      P=0, R2[A] is AVMEMS, R3[A] is current end of VER$
28                **      string
29                **
30                ** Exit:
31                **      P=0, XM set, R3 updated to new location
32                **
33                ** Calls:      None
34                **
35                ** Uses.......
36                **  Inclusive: A[W],C[W],D1,R3[A]
37                **
38                ** Stk lvls:   0
39                **
40                ** History:
41                **
42                **    Date      Programmer            Modification
43                **   --------   ----------    --------------------------------
44                **   10/20/83      NZ         Changed first instruction from
45                **                            CR3EX to C=R3 to fix bug with
46                **                            insuffient memory for my response
47                **                            destroying R3 pointer
48                **   03/30/83      NZ         Changed to just RTNSXM (carry=?)
49                **   11/22/82      NZ         Added code and documentation
50                **
51                ****************************************************************
52                ****************************************************************
53 F511B 11B   =hVER$  C=R3                    Get D1 pointer
54 F511E 135          D1=C                     Put in D1
55 F5121 112          A=R2                     Get AVMEME
56 F5124 1CF          D1=D1- 16                Subtract length I'm adding
57 F5127 137          CD1EX                    Now check if there is room!
58 F512A 8B6          ?A>C   A
59 F512D 42           GOYES  hVER$1            No room...clear carry, exit
60 F512F 135          D1=C                     Room...update D1, R3
61 F5132 10B          R3=C
62 F5135 3F02         LCASC  \ HPIL:  \        (Last 2 filled in by PILVER)
         02A3
         C494
         0584
         02
63 F5147 3300         LC(4)  =PILVER
         00
```

```
 64 F514D 1557          DAT1=C W              Write it out!
 65 F5151       hVER$1
 66 F5151 00            RTNSXM                Set XM (say not handled)
 67              ********************************************************************
 68              ********************************************************************
 69              **
 70              ** Name:       hFINDF - Find file handler (pFINDF poll)
 71              **
 72              ** Category:   POLL
 73              **
 74              ** Purpose:
 75              **      Handle the POLL of (pFINDF), find a specified file
 76              **      in the given mass memory device for HPIL devices
 77              **
 78              ** Entry:
 79              **      R0: First 8 chars of file name
 80              **      R1[3:0]: Last 2 chars of file name
 81              **      D[A]: Device address as returned from FILSPx handler
 82              **      D[S]: Device type from FILSPx
 83              **
 84              ** Exit:
 85              **      Carry clear: (file found, no errors)
 86              **        R0[3:0]: starting record number
 87              **        R0[6:4]: device address
 88              **        R0[10:7]: 0000
 89              **        R0[14:11]: file type
 90              **        R0[15]: 8 (HPIL)
 91              **        R1[0]: entry # in the directory record (0-7)
 92              **        R1[3:1]: record # of the directory entry
 93              **        R1[5:4]: 00
 94              **        R1[9:6]: length of file in sectors
 95              **      Carry set:
 96              **        Error (C[3:0] are the error number)
 97              **
 98              ** Calls:      CKBITL,START,FINDFx,CSLC5,DATSTR,ENDTAP,<ERROR>
 99              **
100              ** Uses:
101              **  Exclusive:     C,        R0,R1,      P
102              **  Inclusive: A,B,C,D[15:5],R0,R1,D0,D1,P,SCRTCH[63:0],ST[5:0]
103              **
104              ** Stk lvls:   6 (FINDFx)
105              **
106              ** History:
107              **
108              **    Date      Programmer            Modification
109              **  --------    ----------    ---------------------------------
110              **  10/14/83      NZ          Updated documentation
111              **  04/01/83      SC          Wrote routine
112              **
113              ********************************************************************
114              ********************************************************************
115 F5153 7D26 =hFINDF GOSUB  CKBITL          Check if HPIL and mass memory
116 F5157 500          RTNNC                 No...don't handle (XM set by CKBITL)
117 F515A 7DC2         GOSUB  Start          Set up the loop, DO
118 F515E 4E2          GOC    hFNFer         Error!
```

```
119 F5161 8E00          GOSUBL =FINDFx      Find the file on the device
          00
120 F5167 452           GOC    hFNFer       Error (either not found or loop err)
121               *
122               * If no carry, then C[3:0] is number of records, B[3:0] is the
123               * directory pointer for the file, A[3:0] is then starting record
124               * of the file on the device, and D1 points to the file type in
125               * the directory entry (which is in SCRTCH[63:0])
126               *
127 F516A 7143          GOSUB  Cslc5        C[8:5]=number of records
128 F516E D2            C=0    A
129 F5170 BF2           CSL    W            C[9:6]=number of records
130 F5173 23            P=     3
131 F5175 A99           C=B    WP           C[3:0]=directory pointer for file
132 F5178 20            P=     0
133 F517A 109           R1=C                R1 is set up for exit conditions
134 F517D 7010          GOSUB  DATSTR       Set up R0 exit conditions in C[W]
135 F5181 108           R0=C                Put into R0 for exit
136 F5184 8EE9          GOSUBL Endtap       Rewind device, unaddress all
          A0
137 F518A 500           RTNNC               If carry clear, done!
138 F518D 6250  hFNFer  GOTO   ERror        Error...set up C[3:0], RTNSC
139               ********************************************************************
140               ********************************************************************
141               **
142               ** Name:       DATSTR, DATST+ - Set up data from FINDFx in C[W]
143               **
144               ** Category:   LOCAL
145               **
146               ** Purpose:
147               **      Set up the data from FINDFx for single register return
148               **
149               ** Entry:
150               **      DATSTR:
151               **        P=0
152               **        D1 points to the file type in RAM (high byte first)
153               **      DATST+:
154               **        D[X]=device address
155               **        A[3:0]=file start address (record number)
156               **
157               ** Exit:
158               **      P=0
159               **      Carry clear
160               **      C[15]=8, C[14:11]=file type, C[6:4]=device address,
161               **      C[3:0]=file start record number
162               **
163               ** Calls:      GT2BYT,CSLC7,CSLC4
164               **
165               ** Uses.......
166               **  Exclusive: C[W],   P
167               **  Inclusive: C[W],D1,P
168               **
169               ** Stk lvls:   1 (GT2BYT)(CSLC7)(CSLC4)
170               **
171               ** History:
```

```
172                **
173                **    Date     Programmer            Modification
174                **  --------   ----------   --------------------------------
175                **  10/14/83      NZ        Added documentation
176                **
177                ****************************************************************
178                ****************************************************************
179 F5191 AF2   DATSTR C=0    W
180 F5194 308          LC(1)  8             Will end up in C[S] (HPIL device)
181 F5197 8E4C         GOSUBL Gt2byt        Read file type from SCRTCH
          A0
182 F519D 8E00         GOSUBL =CSLC7        C[11]=8, C[10:7]=file type
          00
183 F51A3 ABB   DATST+ C=D    X             C[6:3]=0000, C[X]=device address
184 F51A6 7803         GOSUB  Cslc4
185 F51AA 23           P=     3
186 F51AC A96          C=A    WP            Copy file start addr from A[3:0]
187 F51AF 20           P=     0
188 F51B1 03           RTNCC
189                ****************************************************************
190                ****************************************************************
191                **
192                ** Name:       hCREAT - Handle POLL for pCREAT (HPIL device)
193                **
194                ** Category:   POLL
195                **
196                ** Purpose:
197                **      Creates a new file in a mass memory device
198                **
199                ** Entry:
200                **      D[X]=device address
201                **      D[S]=device type (if HPIL, 8)
202                **      STMTR0=first 8 chars of the file name
203                **      STMTR1[3:0]=last 2 chars of the file name
204                **      STMTR1[6:5]=offset to data (from file type table)
205                **      STMTR1[13:10]=file type
206                **      STMTR1[14]=create code
207                **
208                **      R2[A]=first parameter for CREATE:
209                **      ---------------------------------------------------
210                **      Code   Format Implied       Meaning of this parameter
211                **      ----   ----------------     -------------------------
212                **        0    Executable           Data length in nibbles
213                **        1    DATA (fixed length)  Number of records
214                **        2    SDATA (41C data)     Number of (8-byte) registers
215                **        4    TEXT (variable len)  File length in bytes
216                **        8    External type        File length in bytes
217                **
218                **      R3[A]=second parameter for CREATE:
219                **      ---------------------------------------------------
220                **      Code   Format Implied       Meaning of this parameter
221                **      ----   ----------------     -------------------------
222                **        1    DATA (fixed length)  Record length in bytes
223                **      (any)  (not DATA)           (ignored)
224                **
```

```
225                ** Exit:
226                **      P=0
227                **      Carry clear:
228                **        File created on device, initialized if copy code#0
229                **        R3[7:4]=start of data area for file
230                **        R3[15:12]=directory entry pointer for the file
231                **      Carry set:
232                **        Error (C[3:0] is the error number)
233                **
234                ** Calls:      CKHPIL,START,CHKMAS,ASLC3,CSRC4,CSLC4,A-MULT,
235                **              CSLC6,NEWFIL
236                **
237                ** Uses:
238                **  Exclusive: A,  C,  R0-R4,   D1,P,              ST[8]
239                **  Inclusive: A,B,C,D,R0-R4,D0,D1,P,SCRTCH[63:0],ST[8,4:0]
240                **
241                ** Stk lvls:   5 (NEWFIL) (File does not exist currently)
242                **
243                ** History:
244                **
245                **    Date      Programmer             Modification
246                **  --------    ----------    --------------------------------
247                **  10/14/83       NZ         Updated documentation
248                **  04/01/83       SC         Wrote routine
249                **
250                ********************************************************************
251                ********************************************************************
252 F51B3          =hCREAT
253 F51B3 76D5             GOSUB  CKHPIL          Check if device=8
254 F51B7 500              RTNNC                  Not HPIL
255 F51BA 7D62             GOSUB  Start           Set up mailbox, etc
256 F51BE 412              GOC    ERror           Error starting up
257 F51C1 96B              ?D=0   B               Is this LOOP or NULL?
258 F51C4 B0               GOYES  CRTF00          Yes...exit, don't handle
259 F51C6 8E00             GOSUBL =CHKMAS         Check acc ID
            00
260 F51CC 560              GONC   CRTF01          Filbert...continue
261 F51CF 6DE2 CRTF00      GOTO   hCPYXM          Not Filbert...don't handle!
262                *-
263                *-
264 F51D3 AF0  CRTF01      A=0    W
265 F51D6 11A              C=R2
266 F51D9 8AE              ?C#0   A               File size specified?
267 F51DC 80               GOYES  CRTF05          If so, continue
268 F51DE 20               P=     =eRANGE         Error...file size not specified
269 F51E0 6344 ERror       GOTO   hCPYer          Jump to "GOTO   Error"
270                *-
271                *-
272 F51E4 DA   CRTF05      A=C    A               A= First parm (# sectors/bytes)
273 F51E6 1F00             D1=(5) (=STMTR1)+5     Position to offset to data
            000
274 F51ED D2               C=0    A
275 F51EF 14F              C=DAT1 B               C[A] = Offset to data
276 F51F2 137              CD1EX                  Subtract 5 (length of length field)
277 F51F5 1C4              D1=D1- 5
```

```
278 F51F8 137          CD1EX
279 F51FB 178          D1=D1+ 9
280 F51FE 1574         C=DAT1 S              C[S] = Create code of the file
281             *
282             * Look at the create code and implementation field to compute
283             * the total file length in bytes
284             *
285 F5202 94E          ?C#0   S
286 F5205 01           GOYES  CRTF10
287             *
288             * Mainframe executable file (Create code zero)
289             *
290 F5207 CA           A=A+C  A              Length is requested + subheader
291             *
292             * Implementation field for create code zero is length in nibs
293             *
294 F5209 102          R2=A                  Implementation field in directory
295 F520C E4           A=A+1  A
296 F520E 81C          ASRB                  Convert to bytes (round up)
297 F5211 6860 CRTF4.  GOTO   CRTF40         Continue create (A[A] is # bytes)
298             *-
299             *-
300 F5215 A46  CRTF10  C=C+C  S
301 F5218 48F          GOC    CRTF4.         Create code 8...R2, R3 are set up
302 F521B A46          C=C+C  S
303 F521E 5B0          GONC   CRTF20         Not create code 4 (check further)
304             *
305             * Variable length record file (LIF1 type) (Create code 4)
306             *
307 F5221 AF2          C=0    W
308 F5224 10A          R2=C                  Implementation field in directory
309 F5227 425          GOC    CRTF40         Go always (A[A] is # bytes)
310             *-
311             *-
312 F522A A46  CRTF20  C=C+C  S
313 F522D 542          GONC   CRTF30         Not create code 2...must be 1
314             *
315             * HP41C data file
316             *
317             * The bytes of the implementation field in the directory are:
318             *
319             *  BYTE #      MEANING
320             *  -------     ----------------------------------
321             *   28         High order byte of size (in registers)
322             *   29         Low order byte of size
323             *   30         Protection field (0=unsecured, 1=secured)
324             *   31         Unused
325             *
326 F5230 AF2          C=0    W
327 F5233 AE6          C=A    B
328 F5236 F2           CSL    A
329 F5238 F2           CSL    A
330 F523A 814          ASRC
331 F523D 814          ASRC
332 F5240 AE6          C=A    B
```

```
    333 F5243 10A          R2=C                Implementation field in directory
    334 F5246 8E00         GOSUBL =ASLC3       A[A]=file length in nibbles
              00
    335 F524C 81C          ASRB                A[A]=file length in bytes
    336 F524F 4A2          GOC  CRTF40         Go always
    337             *-
    338             *-
    339             *
    340             * FIXED LENGTH RECORD DATA FILE
    341             *
    342 F5252 AF6  CRTF30  C=A    W            C[3:0]= # of logical records
    343 F5255 8E00         GOSUBL =CSRC4
              00
    344 F525B 113          A=R3
    345 F525E 8AC          ?A#0   A            Logical record length specified?
    346 F5261 50           GOYES  CRTF35       Yes...use it
    347 F5263 B24          A=A+1  XS           No...default to 256 bytes
    348 F5266 23   CRTF35  P=     3
    349 F5268 A96          C=A    WP           C[3:0]=Logical record length
    350 F526B 20           P=     0
    351 F526D 7142         GOSUB  Cslc4
    352 F5271 12A          CR2EX               R2[7:4]=Rec length,R2[3:0]=# recs
    353 F5274 8E00         GOSUBL =A-MULT      Compute file length
              00
    354             *
    355             * Now R2 = implementation field, A[A] = file length in bytes
    356             * Put the file size, file type and create code into R1
    357             * Put the file name into R0 and R4[15:12]
    358             *
    359 F527A 1C3  CRTF40  D1=D1- 4
    360 F527D AF2          C=0    W
    361 F5280 15F3         C=DAT1 4            C[3:0] = file type
    362 F5284 8E00         GOSUBL =CSLC6       (into C[9:6])
              00
    363 F528A 25           P=     5
    364 F528C A96          C=A    WP           Copy all 6 nibs
    365 F528F 173          D1=D1+ 4
    366 F5292 1574         C=DAT1 S            C[S] = Create code
    367 F5296 109          R1=C
    368 F5299 AF2          C=0    W
    369 F529C 1CD          D1=D1- 14
    370 F529F 15F3         C=DAT1 4            C[3:0] = Last 2 chars of filename
    371 F52A3 8E00         GOSUBL =CSRC4
              00
    372 F52A9 10C          R4=C                R4[15:12]= Last 2 chars of name
    373 F52AC 1CF          D1=D1- 16
    374 F52AF 1577         C=DAT1 W
    375 F52B3 108          R0=C                R0=First 8 characters of filename
    376 F52B6 840          ST=0   =sOVERW      Do NOT overwrite an existing file!
    377 F52B9 7000         GOSUB  =NEWFIL      Create the file on the tape
    378 F52BD 500          RTNNC               If no carry, no error...done
    379 F52C0 6363         GOTO   hCPYer       Error...set it up
    380             ************************************************************
    381             ************************************************************
    382             **
```

```
                ** Name:       hRDCBF - Read current record into FIB buffer
                **
                ** Category:   POLL
                **
                ** Purpose:
                **      Read the current record of the FIB pointed to by STMTD1
                **      into its FIB buffer
                **      (FAST POLL)
                **
                ** Entry:
                **      STMTD1 contains the FIB address for the file
                **
                ** Exit:
                **      Carry clear, XM=0
                **      Current record has been read into FIB buffer
                **      A[W],D[W], and D1 are restored from SNAPBF (SNAPRS)
                **      If error, jumps directly to BSERR after setting up error
                **
                ** Calls:      STBUF+,START,WRTADR,READR#,CSLC9,UTLEND,ACES=0,
                **             <SNAPRS>,<ERRORX>
                **
                ** Uses:
                **  Inclusive: A,B,C,D,D0,D1,P,ST[4:0]
                **
                ** Stk lvls:   3 (READR#)(START) {1 level saved during these}
                **
                ** Detail:
                **      STBUF+ saves a stack level in D[11:7], RSTORE restores
                **      it to the RSTK
                **
                ** History:
                **
                **    Date     Programmer              Modification
                **  --------   ----------    ---------------------------------
                **  10/14/83      NZ         Updated documentation
                **  04/01/83      SC         Wrote routine
                **
                ****************************************************************
                ****************************************************************
F52C4 7CC0 =hRDCBF GOSUB  STBUF+        Check if HPIL,set up D[X],D1;
                *                       save RSTK level in D[11:7] if HPIL
F52C8 500          RTNNC                Not HPIL
F52CB 7C51         GOSUB  Start         Set up the mailbox, D0
F52CF 4C2          GOC    Errorx        Error exit
F52D2 7B31         GOSUB  WRTADR        Compute current record
                *
                * A[3:0] is the record number of the current record
                *
F52D6 8E00         GOSUBL =READR#       Read the record to the buffer @ D1
      00
F52DC 4F1   RSTOR+ GOC    Errorx        Error exit
                *
                * Restore the RSTK saved in D[11:7]
                * Set access mode in FIB to zero (not modified)
                * Exit through SNAPRS to restore A,D,D0, and D1 from SNAPSV
```

```
437                *
438 F52DF AFB   RSTORE C=D    W              Restore one RSTK level...
439 F52E2 8E00         GOSUBL =CSLC9         ...from D[11:7]
          00
440 F52E8 06           RSTK=C
441                *
442 F52EA 7000         GOSUB  =Utlend        Clean up the loop (unaddress all)
443 F52EE 4D0          GOC    Errorx         Error exit
444 F52F1 7D00         GOSUB  ACES=0         Set access code to zero (clean)
445 F52F5 8D00 =sNAPRS GOVLNG =SNAPRS        Restore A, D, D0, D1
          000
446                *-
447                *-
448 F52FC 8C00 Errorx  GOLONG =ERRORX
          00
449                *-
450                *-
451                *
452                * ACES=0 sets the access code of the current FIB to zero
453                *
454 F5302 7B21  ACES=0 GOSUB  d0=FIB
455 F5306 16A          D0=D0+ =oACCSb        Position to the access code in FIB
456 F5309 D2           C=0    A
457 F530B 15C0         DAT0=C 1              Write out a zero (not modified)
458 F530F 6F64         GOTO   RtnXM0         Return with XM=0, carry clear
459                ****************************************************************
460                ****************************************************************
461                **
462                ** Name:       hWRCBF - Write current record to mass mem device
463                **
464                ** Category:   POLL
465                **
466                ** Purpose:
467                **      Flush the current record for this FIB entry out to
468                **      the mass memory device (buffer contents, current
469                **      position, and record address are not changed by this
470                **      operation)
471                **      (FAST POLL)
472                **
473                ** Entry:
474                **      ST#TD1 contains the FIB address for the file
475                **
476                ** Exit:
477                **      Carry clear, XM=0
478                **      Current record has been flushed out to mass mem device
479                **      A[W],D[W],D0, and D1 are restored from SNAPSV (SNAPRS)
480                **      If error, jumps directly to BSERR after setting up error
481                **
482                ** Calls:      STBUF+,START,WRTADR,WRITE#,<RSTOR+>
483                **
484                ** Uses:
485                **  Inclusive: A,B,C,D,D0,D1,P,ST[8,4:0]
486                **
487                ** Stk lvls:   3 (START)(WRITE#) {a level is saved in D for these
488                **
```

```
                ** History:
                **
                **    Date     Programmer          Modification
                **  --------   ----------   --------------------------------
                **  10/14/83      NZ        Updated documentation
                **  04/01/83      SC        Wrote routine
                **
                ****************************************************************
                ****************************************************************
F5313 7D70      =hWRCBF GOSUB  STBUF+        Check if HPIL, set up D[X],D1;
                *                            save RSTK level in D[11:7] if HPIL
F5317 500               RTNNC                Not HPIL
F531A 7D01              GOSUB  Start         Set up the mailbox, D0
F531E 4DD               GOC    Errorx        Error exit
F5321 7CE0              GOSUB  WRTADR        Compute current record
F5325 8E00              GOSUBL =WRITE#       Write the buffer to mass mem device
      00
F532B 60BF              GOTO   RSTOR+        Restore RSTK level, exit
                ****************************************************************
                ****************************************************************
                **
                ** Name:       hRDNBF - Flush current FIB buffer, read next
                **
                ** Category:   POLL
                **
                ** Purpose:
                **      Flush the current FIB buffer out to the mass memory
                **      device (if altered), read the next record into the FIB
                **      buffer, and update the current position
                **      (FAST POLL)
                **
                ** Entry:
                **      ST=TD1 contains the FIB address for the file
                **
                ** Exit:
                **      Successful:
                **        Carry clear, XM=0
                **        Next record is read into the FIB buffer
                **        Current position in FIB is set to start of next record
                **        File access nibble in FIB is set to zero
                **      Error:
                **        Direct jump to BSERR after setting up the error code
                **
                ** Calls:      STBUF+,START,WRTADR,WRITE#,STUPBF,GETMBX,READR#,
                **             ACES=0,<RSTORE>
                **
                ** Uses:
                **  Inclusive: A,B,C,D,D0,D1,P,ST[8,4:0]
                **
                ** Stk lvls:   3 (START)(WRITE#)(READR#) {1 level saved in D}
                **
                ** History:
                **
                **    Date     Programmer          Modification
                **  --------   ----------   --------------------------------
```

```
                **  10/14/83      NZ          Updated documentation
                **  04/01/83      SC          Wrote routine
                **
                ****************************************************************
                ****************************************************************
F532F 7160 =hRDNBF GOSUB  STBUF+          Check if HPIL, set up D[X], D0;
                *                         save RSTK level in D[11:7] if HPIL
F5333 500          RTNNC                  Not HPIL
F5336 71F0         GOSUB  Start           Set up mailbox, D0
F533A 41C          GOC    Errorx          Error exit
F533D 70D0         GOSUB  WRTADR          Compute current record
F5341 2C           P=     12              Check access (set up by STBUF+)
F5343 90B          ?D=0   P               Is access nibble = 0?
F5346 D0           GOYES  RDNB10          Yes...just read next record
F5348 20           P=     0               No...
F534A 8E00         GOSUBL =WRITE#         Write FIB buffer to mass memory
      00
F5350 4BA          GOC    Errorx          Error exit
                *
F5353 20    RDNB10 P=     0
F5355 7E50         GOSUB  STUPBF          Set up D1 to start of FIB buffer
F5359 78C0         GOSUB  Getmbx          Set D0 back to mailbox
F535D E4           A=A+1  A               Select next record...
F535F 8E00         GOSUBL =READR#         ...read next record
      00
F5365 469          GOC    Errorx          Error exit
F5368 769F         GOSUB  ACES=0          Set access code=0 (not modified)
F536C 16E          D0=D0+ (oDBEGb)-(oACCSb)+5
F536F 16D          D0=D0+ (oCPOSb)-(oDBEGb)-5  Position to current position
F5372 146          C=DAT0 A               Read current position into C[A]
                *
                * The current position is the number of nibbles from data start
                * for the file
                *
F5375 81E          CSRB                   Turn nibbles into bytes (forces
F5378 816          CSRC                     the current position to be at an
F537B 816          CSRC                     even byte boundary when done)
F537E E6           C=C+1  A               Increment to next record number
F5380 812          CSLC
F5383 812          CSLC
F5386 A76          C=C+C  W               Convert back to nibbles
F5389 144          DAT0=C A               Write out updated current position
F538C 7590         GOSUB  Getmbx          Set D0 back to the mailbox
F5390 6E4F         GOTO   RSTORE          Clean up the loop, restore A,D,D0,D1
                ****************************************************************
                ****************************************************************
                **
                ** Name:       STBUF+,STUPBF - Set to read/write current recrd
                ** Name:       WRTADR - Write device addr into FIB, <STUPBF>
                **
                ** Category:   LOCAL
                **
                ** Purpose:
                **      STBUF+:
                **        Check if HPIL...if not, RTNCC,XM=1
```

```
596                **        Save one RSTK level in D[11:7]
597                **      STUPBF:
598                **        Set D[12] to the access nibble for buffer, D1 to the
599                **        FIB buffer, D[A] to the device address, A[3:0] to
600                **        the current record position
601                **
602                ** Entry:
603                **      STMTD1 contains the FIB address of this file
604                **
605                ** Exit:
606                **      Carry clear:
607                **        Not HPIL...XM=1
608                **      Carry set:
609                **        D[12] is the access nibble for this buffer
610                **        D[11:7] is the RSTK value of the caller's caller
611                **        P=0
612                **        D[A] is the device address
613                **        A[3:0] is the current record number
614                **
615                ** Calls:       D0=FIB,CKHPI+,CSRC9,I/OFND,CHKASN
616                **
617                ** Uses.......
618                **  Inclusive: A[W],B[W],C[W],D[W],D0,D1,P
619                **
620                ** Stk lvls:   2 (CHKASN) {RSTK level already saved for this}
621                **
622                ** History:
623                **
624                **    Date      Programmer              Modification
625                **  --------    ----------    ---------------------------------
626                **  10/14/83       NZ         Added documentation
627                **
628                ***************************************************************
629                ***************************************************************
630 F5394 7990   STBUF+ GOSUB  d0=FIB         Set D0 to the start of the FIB
631 F5398 16B          D0=D0+ =oDEVCb        Skip to device type
632 F539B 1564         C=DAT0 S
633 F539F 7DE3         GOSUB  CKHPI+         Check if HPIL
634 F53A3 500          RTNNC                 No...return, carry clear (XM=1)
635 F53A6 07           C=RSTK                .......
636 F53A8 D7           D=C    A              .......
637 F53AA 07           C=RSTK                .. Save caller's caller RSTK value
638 F53AC 8E00         GOSUBL =CSRC9         .. in D[11:7]
          00
639 F53B2 AFF          CDEX   W              .......
640 F53B5 06           RSTK=C                .......
641                *
642 F53B7 7670   STUPBF GOSUB  d0=FIB         Set D0 at FIB entry
643 F53BB 16A          D0=D0+ =oACCSb        Position to access nibble
644 F53BE 2C           P=     12
645 F53C0 1560         C=DAT0 P              Read access nibble into C[12]...
646 F53C4 A87          D=C    P              ...and save it in D[12]
647 F53C7 20           P=     0
648 F53C9 188          D0=D0- (oACCSb)-(oFBF#b)
649 F53CC 146          C=DAT0 A              Read the FIB buffer number([X])
```

```
650 F53CF 8E00          GOSUBL =i/OFND
          00
651 F53D5 4B0           GOC    STUP10         Found the buffer
652 F53D8 300           LC(1)  =eSYSer        Not found..."System Error" (HPIL)
653 F53DB 20            P=     =ePIL          This is an HPIL message
654 F53DD 6E1F          GOTO   Errorx         Error exit
655               *-
656               *-
657 F53E1 167    STUP10 D0=D0+ (oCOPYb)-(oFBF#b)
658 F53E4 16E           D0=D0+ (oDBEGb)-(oCOPYb)+4
659 F53E7 15E6          C=DAT0 7              C[6:0] is device address info
660 F53EB 8E00          GOSUBL =CHKASN        Set up for START to get the addr
          00
661 F53F1 D7            D=C    A              (Info for START into D[3:0])
662               *
663 F53F3 183    STUP20 D0=D0- 4              Position D0 to data begin
664 F53F6 15A3          A=DAT0 4              A[3:0]=data start record number
665 F53FA 163           D0=D0+ 4
666 F53FD 16E           D0=D0+ (oCPOSb)-(oDBEGb)-4
667 F5400 AF2           C=0    W
668 F5403 146           C=DAT0 A              C[A]=current position (in nibbles)
669 F5406 81E           CSRB                  Convert nibble position to byte
670 F5409 F6            CSR    A
671 F540B F6            CSR    A              C[A] is number of records offset
672 F540D CA            A=A+C  A              A[A] is current record number
673 F540F 02            RTNSC                 Carry set=set up for HPIL
674               *-
675               *-
676 F5411 7C10   WRTADR GOSUB  dO=FIB
677 F5415 16C           D0=D0+ (oFBEGb)
678 F5418 16B           D0=D0+ (oDBEGb)-(oFBEGb)+4
679 F541B DB            C=D    A
680 F541D 1543          DAT0=C X              Store device address into FIB
681 F5421 7ECF          GOSUB  STUP20         Set A[3:0] to current record #
682 F5425 8C00 =Getmbx GOLONG =GETMBX         Set D0 back to the mailbox
          00
683               *-
684               *-
685 F542B 8C00 Start   GOLONG =START
          00
686               *-
687               *-
688 F5431 8D00 dO=FIB  GOVLNG =D0=FIB
          000
689               ****************************************************************
690               ****************************************************************
691               **
692               ** Name:       hPRTCL - Print class poll handler for HPIL
693               **
694               ** Category:   POLL
695               **
696               ** Purpose:
697               **      Respond to the PRINT class poll, if this is "OUTPUT"
698               **      or "PLOT" (and the device is HPIL!)
699               **
```

```
700              ** Entry:
701              **      P=0, HEXMODE, D0 @ =MLFFLG
702              **      If this is HPIL and (OUTPUT or PLOT):
703              **        STMTR1[8:2] is the 7 nibble device specifier
704              **        STMTR1[10:9] is the position for OUTPUT/PLOT
705              **        STMTR1[12:11] is the length for OUTPUT/PLOT
706              **        STMTR0[0] is either OUTPTt or PLOTt
707              **
708              ** Exit:
709              **      Carry clear, P=0, HEXMODE
710              **      XM=0:
711              **        Entry conditions for STMTRx are maintained
712              **        STMTR0[5:1] is the address of PRASCI
713              **        STMTR0[10:6] is the address of STMTR1+9
714              **      XM=1: NOT handled by me!
715              **
716              ** Calls:      TSAVD1,CSLC5,CSLC3,TSAV2C,PRTIS+,TRES2C,CSRC3,CSRC
717              **
718              ** Uses.......
719              **  Exclusive:     C,  D0,P
720              **  Inclusive: A,B,C,D,D0,P,FUNCD0,FUNCD1,FUNCR1
721              **
722              ** Stk lvls:   3 (PRTIS+) {2 RSTK levels saved first}
723              **
724              ** NOTE: Must NOT use D1, status bits!
725              **
726              ** History:
727              **
728              **    Date     Programmer             Modification
729              **  --------   ----------   --------------------------------
730              **  12/15/82      NZ        Added documentation
731              **
732              ****************************************************************
733              ****************************************************************
734 F5438        =hPRTCL
735              *
736              * D0 @ MLFFLG now
737              *
738 F5438 D2             C=0    A
739 F543A 14E            C=DAT0 B              Read MLFFLG, type
740 F543D 80D1           P=C    1              P=type
741 F5441 890            ?P=    =OUTPTt        "OUTPUT" type?
742 F5444 70             GOYES  hPRTCO         Yes...do it!
743 F5446 880            ?P#    =PLOTt         "PLOT" type?
744 F5449 84             GOYES  hPRTXM         No...exit, XM=1
745              *
746              * Need to re-setup the loop now!
747              *
748 F544B 8E00   hPRTCO  GOSUBL =TSAVD1        Save D1 temporarily (restored by
         00
749              *                                 PRTIS+)
750 F5451 1F00           D1=(5) (=STMTR1)+2     ...point to 7 nibble handler...
         000
751              *+
752 F5458 07             C=RSTK                Save 2 RSTK levels,ST in FUNCR1
```

```
753 F545A 7150          GOSUB  Cslc5
754 F545E 07            C=RSTK
755 F5460 8E00          GOSUBL =CSLC3
          00
756 F5466 09            C=ST
757 F5468 8E00          GOSUBL =TSAV2C        (Save in FUNCR1)
          00
758             *+
759 F546E 8E00          GOSUBL =PRTIS+        ...set it all up!...
          00
760             *+
761 F5474 8E00          GOSUBL =TRES2C        Restore RSTK levels before check
          00
762 F547A 0A            ST=C                  Restore status bits
763 F547C 8E00          GOSUBL =CSRC3
          00
764 F5482 06            RSTK=C                Restore second level
765 F5484 8E00          GOSUBL =CSRC5
          00
766 F548A 06            RSTK=C                Restore first level
767             *+
768 F548C 831           ?XM=0                 Handled?
769 F548F 60            GOYES  hPRTC1         Yes...continue
770 F5491 6B20 hPRTXM   GOTO   hCPYXM         No...exit,XM=1,carry clear
771             *-
772             *-
773 F5495       hPRTC1
774             *
775             * Loop is set up now, A[A] is address of PRASCI
776             *
777 F5495 25            P=     5
778 F5497 3400          LC(5)  (=STMTR1)+9    Position and length for OUTPUT
          000
779 F549E 20            P=     0
780 F54A0 1B00          DO=(5) (=STMTR0)+1    Handler address
          000
781 F54A7 D6            C=A    A              Copy handler address from A[A]
782 F54A9 15C9          DATO=C 10             (Write it out!)
783 F54AD 03            RTNCC                 Done!
784             ****************************************************************
785             ****************************************************************
786             **
787             ** Name:       hCOPYx - Copy POLL handler (HPIL)
788             **
789             ** Category:   STEXEC
790             **
791             ** Purpose:
792             **      Handler for COPY execute POLL
793             **
794             ** Entry:
795             **      A[W] is first 8 chars of filename
796             **      R0[3:0] is last 2 chars
797             **      D[A] is source device information
798             **      P=0
799             **      ST(=sEXTDV) set if either of both file specs are HPIL
```

```
800             **      ST(=sUNDEF) set if both file names are zero (undef'd)
801             **      ST(=sCARD) set if destination device is CARD or PCRD
802             **      R2 has destination device info!!!!!
803             **      SAVSTK: (offsets from SAVSTK pointer)
804             **        -62 =>  -1: (POLL save area)
805             **        -87 => -63: (Source info)
806             **       -112 => -88: (Destination info)
807             **
808             **        Info format: low mem        ---          high mem
809             **                     First 8 chars...last 2 chars...device
810             **
811             ** Exit:
812             **      P=0
813             **      Carry set: Error...error # in C[3:0]
814             **      Carry clear:
815             **        XM=0: handled
816             **        XM=1: not handled
817             **      SAVSTK unchanged from entry
818             **
819             ** Calls:      ASLC4;6;12,ASRC3;4;5;10,BLANKC,CHAIN-,CHKBIT,
820             **             CLMODE,CRTF,CSLC2;5;10,CSRC5;10,D0=FR0,D1=S20,
821             **             DdtRd,ENDTAP,FINDF,FINDFL,FNDMB+,FRAME-,GETBYT,
822             **             GETD,GETDev,GETDST,GETMBX,GETTYP,GETX,hCPY5S,
823             **             hCPYE.,hCPYEL,hCPYXM,hRNMsd,LEXBF+,MOVEFL,NEWFI+,
824             **             PRGFMF,PUTE,RDINFD,RDINFO,READSU,SEEKA,TRES2C,
825             **             TSAV2C,TSTAT,UTLEND
826             **
827             ** Uses.......
828             **  Inclusive: A-D,R0-R4,D0,D1,P,ST[8,4:0],FUNCR0;1,FUNCD0,SCRTCH
829             **
830             ** Stk lvls:   6 (NEWFIL;PUGFIB)
831             **
832             ** History:
833             **
834             **    Date     Programmer          Modification
835             **  --------   ----------    --------------------------------
836             **  12/21/83      NZ         Added check for zero-length file
837             **                           in hCPY50...was sending an SDA
838             **                           even if no more data was expected
839             **                           from the device
840             **  10/30/83      NZ         Added fix for bug...if in device
841             **                           mode and receive a zero-length file
842             **                           which already exists in RAM, the
843             **                           machine would lock up.  D0 was
844             **                           being destroyed in the check for
845             **                           the file existing (FINDF).
846             **  09/07/83      NZ         Added check for destination=HPIL
847             **                           for COPY from mainframe to external
848             **  05/12/83      NZ         Removed convert to upper case for
849             **                           destination
850             **  01/12/83      NZ         Updated documentation
851             **
852             ****************************************************************
853             ****************************************************************
854 F54AF 812  Cslc5   CSLC
```

```
855 F54B2 8C00 Cslc4   GOLONG =CSLC4
          00
856             *-
857             *-
858 F54B8       =hCOPYx
859 F54B8 870           ?ST=1  =sEXTDV       Is any of this external device?
860 F54BB 80            GOYES  hCPY10        Yes...continue
861 F54BD       hCPY6.
862             *
863             * Copy tape to tape (whole volume)
864             *
865             * TWO cases...both on same loop vs. on different loops!
866             *
867 F54BD 21    hCPYXM  P=     1
868 F54BF 0D            P=P-1                Clear carry...
869 F54C1 00            RTNSXM               Return, carry clear
870             *-
871             *-
872 F54C3 872   hCPY10  ?ST=1  =sCARD        Is either one CARD?
873 F54C6 7F            GOYES  hCPYXM        Yes...not for me!!!
874 F54C8 DB            C=D    A
875 F54CA B06           C=C+1  P             Check if source is mainframe
876 F54CD 442           GOC    hCPY3.        Source is (not specified)
877 F54D0 DB            C=D    A             Not (not specified)...
878 F54D2 A06           C=C+C  P             Check if source is HPIL
879 F54D5 5C1           GONC   hCPY3.        Source is NOT HPIL...copy main
880             *
881             * Source is external...check if HP-IL
882             *
883 F54D8 90E           ?C#0   P
884 F54DB 2E            GOYES  hCPYXM        Not for me!
885             *
886             * Source IS HP-IL...check further
887             *
888 F54DD 11A           C=R2                 Read back destination info
889 F54E0 D5            B=C    A             Save device in B[A] for loop>loop
890 F54E2 B06           C=C+1  P             Check if dest is (not specified)
891 F54E5 480           GOC    hCPY5.        Destination is (not specified)
892 F54E8 A05           B=B+B  P             Check if destination is HPIL
893 F54EB 451           GOC    hCPY12        Destination is external
894 F54EE 62B2 hCPY5.   GOTO   hCPY50        Destination is not HPIL
895             *-
896             *-
897 F54F2       hCPY3.
898             *
899             * Check if destination is HPIL
900             *
901 F54F2 11A           C=R2
902 F54F5 A06           C=C+C  P             This MUST carry...sEXTDV was set
903 F54F8 90E           ?C#0   P             Is this HPIL?
904 F54FB 2C            GOYES  hCPYXM        No...don't handle!
905 F54FD 6A21          GOTO   hCPY30        Copy from main to loop
906             *-
907             *-
908 F5501       hCPY12
```

```
909                *
910                * Destination is external...check if HP-IL
911                *
912 F5501 90D              ?B#0   P
913 F5504 9B               GOYES  hCPYXM        Not HP-IL!
914                *
915                * Source, destination are both HPIL...check if name given
916                *
917 F5506 871              ?ST=1  =sUNDEF       Names undefined?
918 F5509 4B               GOYES  hCPY6.        Yes...copy tape to tape
919                *
920                * Named HPIL to HPIL transfer
921                *
922 F550B F7               DSR    A             Shift address into D[X]
923                *
924                * Copy a file from HPIL to HPIL (may be same device)
925                *
926                * first find the source file
927                *
928 F550D 8E00              GOSUBL =FINDFL       Find the source file
           00
929 F5513 560              GONC   hCPY22        OK...continue
930 F5516 6392             GOTO   hCPY5?        Error...set it, return!
931                *-
932                *-
933                *
934                * Now save starting sector, etc in R3
935                *
936 F551A 8E00 hCPY22      GOSUBL =CSRC5        Temp put # sectors in C[15:11]
           00
937 F5520 D6               C=A    A             Copy starting sector to C[A]
938 F5522 74B6             GOSUB  Cslc10        Put # of sectors in C[9:5]
939 F5526 DB               C=D    A             Copy device address to C[A]
940 F5528 10B              R3=C                 Save all in R3!
941                *
942                * Now R3[A] is device address, [9:5] is # sectors, [14:10] is
943                * first sector address
944                *
945                * Now check the file type for private, copy code, unknown, etc
946                *
947 F552B 7B77             GOSUB  GETTYP        Read in file type & check it
948 F552F 460              GOC    hCPY23        OK...found it!
949 F5532 6CB2 hCPYtP      GOTO   hCPYtp        Illegal (unrecognized) type
950                *-
951                *-
952 F5536      hCPY23
953                *
954                * B[S] is offset into type table, B[A],C[A] point to entry,
955                * A[A] is file type
956                *
957 F5536 135              D1=C                 Set D1 @ table start
958 F5539 A4D              B=B-1  S             Convert to base zero entry
959 F553C AC2              C=0    S
960 F553F B46              C=C+1  S             C[S] is now max non-private type
961 F5542 B49              C=C-B  S             If carry, then private
```

```
 962 F5545 560          GONC   hCPY24        OK...not private
 963 F5548 6997         GOTO   hPURSC        Illegal type (private)
 964             *-
 965             *-
 966             *
 967             * Type is acceptable to copy!!
 968             *
 969             * (Chose the FIRST type...not secure or private)
 970             *
 971 F554C 17F  hCPY24  D1=D1+ 16            Point to UN type
 972 F554F 15B3         A=DAT1 4            Read the type
 973 F5553 7C37         GOSUB  D1=S20       Position to TYPE in SCRTCH
 974 F5557 1593         DAT1=A 4            Write it out for now
 975             *
 976             * Now set up with destination name, etc for NEWFIL
 977             *
 978 F555B 8E88          GOSUBL hRNMsd       Read dest, convert to UC, etc
          80
 979             *
 980             * Now A[W], R0[3:0] is filename, D[A] is unchanged, R1 is dest
 981             * address
 982             *
 983             * Check if destination device is (not specified) or (HPIL)
 984             *
 985 F5561 119          C=R1
 986 F5564 816          CSRC                Rotate into C[S], [X]
 987 F5567 B46          C=C+1  S
 988 F556A 440          GOC    hCPY25       Not specified...go on!
 989             *
 990             * Address is specified...put it in D!
 991             * (To get here, destination address had to be specified)
 992             *
 993 F556D D7           D=C    A
 994 F556F 120  hCPY25  AROEX               Put first 8 chars in R0...
 995 F5572 8E00          GOSUBL =ASRC4       ...move last two to A[15:12]...
          00
 996 F5578 104          R4=A                ...and put in R4[15:12]!
 997             *
 998             * New name is now set up...set up type and size
 999             *
1000 F557B 7417         GOSUB  D1=S20       Point to file type
1001 F557F AF2          C=0    W            Preclear high nibbles!
1002 F5582 15F3         C=DAT1 4            Read the type (written above!)
1003 F5586 113          A=R3                Get back # sectors to A[9:5]
1004            *
1005            * SOURCE info:
1006            *
1007            * A[A] is device addr, A[9:5] is # sectors, A[14:10] is sector
1008            *   address of data
1009            *
1010 F5589 7F07         GOSUB  D0=FR0       Set D0=FUNCR0
1011 F558D 1507         DAT0=A W            Save R3 contents in FUNCR0
1012 F5591 8E00         GOSUBL =ASRC5       # sectors to A[A]
          00
1013 F5597 741F         GOSUB  Cslc5        File type to C[8:5]
```

```
1014 F559B D6          C=A    A            Copy # sectors
1015 F559D F2          CSL    A            # sectors*16
1016 F559F BF2         CSL    W            # sectors*256 (# bytes) in C[5:0]
1017 F55A2 109         R1=C                R1 is now set up for NEWFIL!
1018 F55A5 1D00        D1=(2) (=SCRTCH)+56 Point to implementation bytes
1019 F55A9 15F7        C=DAT1 8
1020 F55AD 10A         R2=C                R2 is set up for NEWFIL
1021 F55B0 DB          C=D    A            Copy address to C[A]
1022 F55B2 8E00        GOSUBL =TSAV2C      Save source address in STMTR1
           00
1023               *
1024               * Now set up to call NEWFIL to create the file
1025               *
1026 F55B8 840         ST=0   =sOVERW      Do NOT overwrite the file!
1027 F55BB 8E00        GOSUBL =NEWFI+      START, Create the file
           00
1028 F55C1 426         GOC    hCPYer       Error
1029               *
1030               * Now R3 is B[W] contents from NEWFIL, FUNCR1 is unchanged
1031               *
1032 F55C4 8E00        GOSUBL =TRES2C      Restore source address to C[A]
           00
1033 F55CA 109         R1=C                Store address in dest field
1034 F55CD 7BC6        GOSUB  DO=FRO       Set DO to FUNCRO
1035 F55D1 1567        C=DATO W            Recall source file info to C[W]
1036 F55D5 10A         R2=C                Store address in source field
1037 F55D8 7B86        GOSUB  Csrc10       Get source sector addr to C[A]
1038 F55DC D5          B=C    A            Sector address of source
1039 F55DE 113         A=R3
1040 F55E1 8E00        GOSUBL =ASRC3       Get file start into A[4:1]
           00
1041 F55E7 F4          ASR    A            (Clear high nibble of A[A])
1042 F55E9 11A         C=R2                Recall # of sectors to C[9:5]
1043 F55EC D6          C=A    A            Get sector # of destination
1044 F55EE 10B         R3=C                R3 is now set up for MOVEFL
1045               *
1046               * Now set up for MOVEFL
1047               *
1048 F55F1 8E00        GOSUBL =MOVEFL      Move the file between devices
           00
1049 F55F7 4C2         GOC    hCPYer       Error
1050               *
1051               * Now clean up the tape(s) (rewind, etc)
1052               *
1053 F55FA 11A         C=R2                Get source addr from R2[A]
1054 F55FD D7   hCPY28 D=C    A            Save in D[A]
1055 F55FF 8E00        GOSUBL =CHKBIT      Check if Filbert tape
           00
1056 F5605 521         GONC   hCPY29       Not a Filbert...try next device
1057 F5608 8E00        GOSUBL =FNDMB+      Find that mailbox
           00
1058 F560E 451         GOC    hCPYer       Error if carry
1059 F5611 7316        GOSUB  Endtap       Filbert...clean up (rewind,etc)
1060 F5615 4E0         GOC    hCPYer       Error if carry
1061 F5618       hCPY29
```

```
1062 F5618 119          C=R1                Get dest addr from R1[A]
1063 F561B 937          ?C#D   X            Is this a new device?(addr,loop#)
1064 F561E FD           GOYES  hCPY28        Yes...clean it up also
1065 F5620 6E51         GOTO   RtnXM0        Done...exit
1066             *-
1067             *-
1068 F5624 6296 hCPYer  GOTO   Error         Error...set C[3:0] to code
1069             *-
1070             *-
1071 F5628      hCPY30
1072             *
1073             * Code to set up mainframe to loop copy
1074             *
1075             * First find the source file in the mainframe
1076             *
1077             * Filename is already in A[W]...shift D[A] around for FINDF
1078             *
1079             * (If filename is undefined i.e. zero, FINDF will error out)
1080             *
1081 F5628 817          DSRC                Put D[0] into D[S]...
1082 F562B 8F00         GOSBVL =FINDF       Find the file
         000
1083 F5632 3300         LC(4)  =eFnFND      File not found
         00
1084 F5638 400          RTNC                Return with error in C[3:0]
1085             *
1086             * D1 points to the start of file now
1087             *
1088             * Get the info about the file and put it in R1-R2
1089             *   (size, type, data start address, implementation bytes)
1090             *
1091 F563B 17F          D1=D1+ =oFTYPh       Skip name
1092 F563E 173          D1=D1+ =oFLAGh       Skip type
1093             *
1094             * Now pointing to the flag field...read protection, copy code
1095             *
1096 F5641 14B          A=DAT1 B            Flags (bit 0=SE,bit 1=PR)
1097 F5644 17B          D1=D1+ (oFLENh)-(oFLAGh) Leave D1 @ file length
1098 F5647 302          LCHEX  2            Privacy bit
1099 F564A 0E02         C=C&A  P            (Could be A for code space)
1100 F564E 90A          ?C=0   P
1101 F5651 60           GOYES  hCPY31       Not private...continue
1102             *
1103             * Attempt to copy a private file...error!
1104             *
1105 F5653 6E86         GOTO   hPURSC       Protection error
1106             *-
1107             *-
1108 F5657      hCPY31
1109             *
1110             * File is legal to copy...check copy code
1111             *
1112 F5657 F4           ASR    A            A[0] is now copy code
1113             *
1114             * Following instruction clears C[A] - used below this!
```

```
1115                *
1116 F5659 D2              C=0     A
1117 F565B A86             C=A     P               Read copy code into C[0]
1118 F565E AF0             A=0     W               Clear A[W]
1119 F5661 143             A=DAT1  A               Pre-read file length into A[W]!
1120 F5664 25              P=      =lFLENh         Skip length of length field
1121 F5666 80F0            CPEX    0
1122 F566A EA              A=A-C   A
1123 F566C 80F0            CPEX    0               Restore copy code
1124 F5670 20              P=      0               Reset P=0
1125                *
1126                * Decode what the copy code is!
1127                *
1128 F5672 90A             ?C=0    P               Is this copy code 0?
1129 F5675 42              GOYES   hCPY33          Yes...do it
1130 F5677 A06             C=C+C   P               Unknown type? (CC=8)
1131 F567A 4A1             GOC     hCPY32          Yes...can't handle it
1132 F567D A06             C=C+C   P               ASCII text file? (CC=4)
1133 F5680 560             GONC    hCPY3a          No...keep checking
1134 F5683 6580            GOTO    hCPY34          Yes...do it
1135                *-
1136                *-
1137 F5687 A06  hCPY3a     C=C+C   P               HP41C data file? (CC=2)
1138 F568A 560             GONC    hCPY3b          No...TITAN data file
1139 F568D 6580            GOTO    hCPY36          Yes...do it
1140                *-
1141                *-
1142                *
1143                * TITAN fixed length data file (CC=1)
1144                *
1145 F5691 6FA0 hCPY3b     GOTO    hCPY38
1146                *-
1147                *-
1148 F5695      hCPY32
1149                *
1150                * Unknown file type...exit or poll?
1151                *
1152 F5695 672E            GOTO    hCPYXM          I give up!
1153                *-
1154                *-
1155 F5699      hCPY33
1156                *
1157                * Mainframe executable file (COPY CODE = 0)
1158                * D1 points to file length field
1159                * A[W] is file length in nibbles (data + subheader)
1160                *
1161 F5699 102             R2=A                    Set implementation bytes<==length
1162                *
1163                * D1 is pointing at FLENh, A[W] is length in bytes, C[A] is 5.
1164                *
1165 F569C 101  hCPY3-     R1=A                    Put file len in nibs in R1[5:0]
1166                *
1167                * Now get actual file start address
1168                *
1169 F569F 305             LC(1)   =lFLENh         Offset to data for mainframe
```

```
1170 F56A2 133  hCPY3+  AD1EX
1171 F56A5 131          D1=A                    Copy D1==>A
1172 F56A8 CA           A=A+C  A                Add offset to data start
1173 F56AA 8E00         GOSUBL =ASLC4           Rotate into A[8:4]
           00
1174                *
1175                * Now get the file type (from the source)
1176                *
1177 F56B0 1CF          D1=D1- (oFLENh)-(oFTYPh) Move to file type
1178 F56B3 15B3         A=DAT1 4                Read it
1179                *
1180                * check if BASIC file...if so, set flag "BASIC"
1181                *
1182                Basic   EQU    0
1183 F56B7 840          ST=0   Basic
1184 F56BA 3341         LC(4)  =fBASIC
           2E
1185 F56C0 23           P=     3
1186 F56C2 916          ?A#C   WP
1187 F56C5 50           GOYES  hCPY3f
1188 F56C7 850          ST=1   Basic            Set Basic flag
1189 F56CA          hCPY3f
1190                *
1191                * Rotate file type into A[9:6], file start into A[14:10]
1192                *
1193 F56CA 8E00         GOSUBL =ASLC6
           00
1194 F56D0 119          C=R1                    Read back the length...
1195 F56D3 E6           C=C+1  A                ...add 1 to round UP...
1196 F56D5 81E          CSRB                    ...convert to bytes!
1197 F56D8 DA           A=C    A                (NOT WP: nibble 5 is always zero)
1198 F56DA 101          R1=A                    Now size, type, and start are set
1199 F56DD 860          ?ST=0  Basic            Is this NOT a BASIC file?
1200 F56E0 52           GOYES  hCPY3g           Not BASIC...continue
1201                *
1202                * This is a BASIC file...chain it first!
1203                *
1204 F56E2 8E00         GOSUBL =ASRC10          File start ==> A[A]
           00
1205 F56E8 20           P=     0
1206 F56EA D2           C=0    A
1207 F56EC 3113         LC(2)  (=oFLENh)+(oBSsod)
1208 F56F0 EA           A=A-C  A
1209                *
1210                * Now A[A] is the start of the file header
1211                *
1212 F56F2 11A          C=R2
1213 F56F5 10B          R3=C                    Save R2 in R3 for now...
1214 F56F8 8F00         GOSBVL =CHAIN-          Chain the file
           000
1215 F56FF 11B          C=R3
1216 F5702 10A          R2=C                    Restore R2 from R3!
1217 F5705 6A50 hCPY3g  GOTO   hCPY39           Get the destination name, do it!
1218                *-
1219                *-
```

```
1220 F5709      hCPY34
1221            *
1222            * Handler for ASCII (TYPE=1) text files (COPY CODE = 4)
1223            *
1224            * D1 points to FLENh, A[W] is file length in nibbles
1225            *
1226 F5709 AF2          C=0    W
1227 F570C 10A          R2=C                  Clear implementation bytes
1228 F570F 6C8F         GOTO   hCPY3-         Continue at common code
1229            *-
1230            *-
1231 F5713      hCPY36
1232            *
1233            * Handler for HP41C data file (COPY CODE = 2)
1234            * D1 points to file length field, A[W] is file length in nibbles
1235            *
1236 F5713 101          R1=A                  Save file length in R1[5:0](nibs)
1237 F5716 AF2          C=0    W              Check if it fits...
1238 F5719 D6           C=A    A              C[W] is file length
1239 F571B 972          ?A=C   W              Contained in [A] field?
1240 F571E 60           GOYES  hCPY37         OK...continue
1241 F5720 68B0         GOTO   hCPY5!         Too big...size error
1242            *-
1243            *-
1244 F5724 8E00 hCPY37  GOSUBL =CSRC5         Rotate high byte to C[15:14]
           00
1245 F572A AB6          C=A    X              Copy low byte (ignore low nibble)
1246 F572D F6           CSR    A
1247 F572F 8E00         GOSUBL =CSLC2         C[B] is high byte, C[3:2] is low
           00
1248 F5735 10A          R2=C                  Set up implementation bytes!
1249 F5738 D2           C=0    A
1250 F573A 305          LC(1)  =o41sod        Offset for 41C data file
1251 F573D 646F         GOTO   hCPY3+
1252            *-
1253            *-
1254 F5741      hCPY38
1255            *
1256            * Handler for fixed length data files (COPY CODE = 1)
1257            * D1 points to file length field, A[W] is file length in nibbles
1258            *
1259 F5741 308          LC(1)  8              Subtract impl bytes from length
1260 F5744 EA           A=A-C  A
1261 F5746 101          R1=A                  Save actual file length in nibs
1262 F5749 174          D1=D1+ =lFLENh        Skip to implementation fields
1263 F574C 15B7         A=DAT1 8              Read them...
1264 F5750 102          R2=A                  Set up implementation field in R2
1265 F5753 111          A=R1                  Get file length back for hCPY3+
1266 F5756 1C4          D1=D1- =lFLENh        Move back to FLENh
1267 F5759 30D          LC(1)  (=lFLENh)+8    Point past implementation field
1268 F575C 654F         GOTO   hCPY3+         Finish up
1269            *-
1270            *-
1271 F5760      hCPY39
1272 F5760 72F4         GOSUB  Rdinfd         Read the info from SAVSTK
```

```
1273                *
1274                * Now A is first 8 chars, R0 is last 2 chars, D is device info
1275                *
1276 F5764 817             DSRC                Shift device info...addr->D[A]
1277 F5767 120             AROEX               Put first 8 chars in R0
1278 F576A 8E00            GOSUBL =ASLC12      Rotate last 2 chars to A[15:12]
          00
1279 F5770 104             R4=A                Now last 2 chars in R4[15:12]
1280                *
1281                * Do the actual transfer now
1282                *
1283                * (Get the mailbox back - NEWFI+ does START, NEWFIL)
1284                *
1285 F5773 850             ST=1   =sOVERW      Allow overwriting existing file
1286 F5776 8E00            GOSUBL =NEWFI+      Create a new file on the tape
          00
1287 F577C 436             GOC    hCPY5a       Error...set it up!
1288 F577F 821  RtnXM0     XM=0                No error...return CC, XM=0
1289 F5782 03              RTNCC
1290                *-
1291                *-
1292 F5784 8E00 =CKBITL    GOSUBL =CHKBIT      Check if bit for Filbert is set
          00
1293 F578A 501             GONC   CKHPIx       No carry...set XM (not Filbert)
1294 F578D ACB  =CKHPIL    C=D    S
1295 F5790 A46  =CKHPI+    C=C+C  S            Check if external
1296 F5793 570             GONC   CKHPIx       Not external...don't handle
1297 F5796 94A             ?C=0   S            HPIL?
1298 F5799 00              RTNYES              Yes...return, set carry
1299 F579B 00   CKHPIx     RTNSXM              Carry clear, XM=1
1300                *-
1301                *-
1302 F579D 6F1D hCPYxM     GOTO   hCPYXM
1303                *-
1304                *-
1305 F57A1      hCPY50
1306                *
1307                * Copy from loop to main
1308                *
1309                * A[W] is first 8 chars, R0[3:0] is last 2 chars
1310                * D[A] is device of source
1311                *
1312 F57A1 817             DSRC                Shift device back to normal
1313 F57A4 8E00            GOSUBL =FINDFL      Save first 8, START,FINDFx
          00
1314 F57AA 06   hCPY5?     RSTK=C              Save (possible) error message
1315                *
1316                * Found the file (A[3:0] is start, C[3:0] is length, D1->type)
1317                * (If this is LOOP, then may have a bad name, but rest is OK)
1318                *
1319 F57AC 7FFC            GOSUB  Cslc5        Save length in [9:5]
1320 F57B0 D6              C=A    A            Start in [A]
1321 F57B2 79FC            GOSUB  Cslc5        Start to [9:5], length to [14:10]
1322 F57B6 10C             R4=C
1323 F57B9 11A             C=R2                C[A] is destination type
```

```
1324 F578C D7          D=C    A
1325 F578E 817         DSRC                  Rotate device into correct place!
1326 F57C1 07          C=RSTK                Restore (possible) error message
1327            *
1328            * Now R4[14:10] is length, R4[9:5] is start,
1329            * D1 points to file type
1330            *
1331 F57C3 502         GONC   hCPY51         Found it!
1332 F57C6 880         ?P#    0              NOT "Device mode error, tape"?
1333 F57C9 71          GOYES  hCPY5a         No...error exit!
1334 F57CB D0          A=0    A              Set A[A]=0 (# left to read)
1335 F57CD 102         R2=A                  Set R2[A]=0
1336 F57D0 300         LC(1)  =eNFILE        "File not found"
1337 F57D3 20          P=     =eTAPE         Set P value for "File not found"
1338 F57D5 6143        GOTO   hCPYeL         Go clean up (or exit)
1339            *-
1340            *-
1341 F57D9 20   hCPY5!  P=     0
1342 F57DB 300         LC(1)  =eTSIZE
1343 F57DE 20   hCPY5t  P=     =eTAPE         Size error!
1344 F57E0 634E hCPY5a  GOTO   hCPYer
1345            *-
1346            *-
1347 F57E4      hCPY51
1348 F57E4 72C4        GOSUB  GETTYP         Read file type & get type entry
1349 F57E8 401         GOC    hCPY52         OK...type in A[A]
1350            *
1351            * Unrecognized type...error
1352            *
1353            * Clean up the loop (check for LOOP or non-MS source)
1354            *
1355 F57EB 7673 hCPYt-  GOSUB  hCPYel         Error...check for copy from loop
1356 F57EF 20   hCPYtp  P=     0              Guarantee P=0 here
1357 F57F1 3300        LC(4)  =eFTYPE        Illegal (Unrecognized) type
          00
1358 F57F7 02          RTNSC
1359            *-
1360            *-
1361 F57F9 D8   hCPY52  B=A    A              Copy file type to B[A]
1362 F57FB 101         R1=A                  Save file type in R1 for CRTF
1363            *
1364            * Now B[S] is position of file type within entry
1365            * C[A] points to start of entry
1366            *
1367 F57FE 135         D1=C                  Set D1 to start of entry
1368 F5801 1574        C=DAT1 S              Read the create code for the file
1369            *
1370 F5805 A4D         B=B-1  S              Convert entry # to base zero
1371 F5808 114         A=R4
1372 F580B AC0         A=0    S
1373 F580E B44         A=A+1  S
1374 F5811 BCC         A=-A-1 S              A[S]="1110" (binary)
1375 F5814 0E40        A=A&B  S              A[S] is new security code (not
1376            *                              secure!)
1377 F5818 104         R4=A                  Save security code in R4[S]
```

```
1378                *
1379                * C[S] is create code for this file.
1380                * B[A] is file type for this file.
1381                *
1382 F581B 1F00       000      D1=(5) (=SCRTCH)+56  Point to implementation bytes
1383 F5822 94A               ?C=0   S             Check if mainframe type
1384 F5825 17                GOYES  hCPY56         Yes...set it up
1385 F5827 A46               C=C+C  S             Check if external...
1386 F582A 560               GONC   hCPY5j        ...no...keep checking
1387 F582D 6AA0              GOTO   hCPY5-        ...yes...will be set up in CRTF
1388                *-
1389                *-
1390 F5831 A46  hCPY5j       C=C+C  S             Check if create type is LIF1
1391 F5834 4C0               GOC    hCPY53        Yes...set it up
1392 F5837 A46               C=C+C  S             Check if type is 41C data file
1393 F583A 454               GOC    hCPY55        Yes...set it up
1394                *                           Type is TITAN data file...
1395 F583D 6820              GOTO   hCPY54        ...set it up
1396                *-
1397                *-
1398                *
1399                * LIF1 file type
1400                *
1401 F5841 23   hCPY53       P=     3
1402 F5843 1D00              D1=(2) (=SCRTCH)+32  Length field
1403 F5847 AF2               C=0    W
1404 F584A 7000              GOSUB  =GETBYT       Read 4 bytes @ length
1405 F584E BF2               CSL    W
1406 F5851 BF2               CSL    W             Convert to BYTES!
1407 F5854 10A               R2=C                 Store in R2
1408                *
1409                * Check if "reasonable" size
1410                *
1411 F5857 D2                C=0    A             Clear low end!
1412 F5859 97A               ?C=0   W             Bigger than 1M bytes?
1413 F585C C7                GOYES  hCPY5-        No...do it!
1414 F585E 7303 hCPY5%       GOSUB  hCPYel        Check for more bytes to read
1415 F5862 667F              GOTO   hCPY5!        Yes...size error
1416                *-
1417                *-
1418 F5866       hCPY54
1419                *
1420                * TITAN data file type
1421                *
1422 F5866 AF0               A=0    W             Clear high nibble first
1423 F5869 173               D1=D1+ 4             Point to record length
1424 F586C 15B3              A=DAT1 4             Read record length...
1425 F5870 103               R3=A                 ...and save in R3
1426 F5873 1C3               D1=D1- 4             Point back to # of records
1427 F5876 15B3              A=DAT1 4             Read # of records
1428 F587A 102  hCPY5b       R2=A                 Put into R2
1429 F587D 5A5               GONC   hCPY5-        Go always...finish it up
1430                *-
1431                *-
```

```
1432 F5880 AF0  hCPY55  A=0    W
1433 F5883 103          R3=A                R3[4] must be zero for CRTF
1434 F5886 14B          A=DAT1 B            Read high byte of size
1435 F5889 F0           ASL    A
1436 F588B F0           ASL    A
1437 F588D 171          D1=D1+ 2
1438 F5890 14B          A=DAT1 B            Read low byte of size
1439 F5893 56E          GONC   hCPY5b       Go always
1440            *-
1441            *-
1442            *
1443            * This is a mainframe create code!
1444            *
1445 F5896      hCPY56                      D1<=start of implementation bytes
1446            *
1447            * First read in offset to data from @ C[A]+3 (set up by FTYPF#)
1448            *
1449 F5896 137          CD1EX
1450 F5899 172          D1=D1+ 3
1451 F589C DA           A=C    A            Start of implementation bytes
1452 F589E AF2          C=0    W
1453 F58A1 14F          C=DAT1 B            Offset to data in C[W]
1454 F58A4 131          D1=A
1455 F58A7 15B5         A=DAT1 6            Read in the file length
1456 F58AB 25           P=     5
1457 F58AD B1A          A=A-C  WP           Subtract off offset to data
1458 F58B0 20           P=     0
1459 F58B2 3150         LC(2)  =lFLENh      Length of file length field
1460 F58B6 25           P=     5
1461 F58B8 A1A          A=A+C  WP           (Add this back to length)
1462            *
1463            * Now A[5:0] contains the length of data portion of the file
1464            *
1465 F58BB 102          R2=A                Save in R2 for future use...
1466 F58BE 90C          ?A#0   P
1467 F58C1 D9           GOYES  hCPY5%       Error...size
1468            *
1469            * Check if this size is reasonable...
1470            *
1471 F58C3 11C          C=R4
1472 F58C6 7D93         GOSUB  Csrc10       Get length into C[A]
1473 F58CA BF2          CSL    W
1474 F58CD BF2          CSL    W            Convert to bytes...
1475 F58D0 A76          C=C+C  W            ...now to nibbles...
1476 F58D3 996          ?A>C   WP           ...check if bigger (corrupt!!!)
1477 F58D6 88           GOYES  hCPY5%       Error...file size
1478            *
1479            * Passed reasonability test
1480            *
1481            * R2 contains # of nibbles for copy code 0, # of logical
1482            * records for other codes; R3 contains the record size in
1483            * bytes (If create code is 8, none of these are defined yet)
1484            *
1485 F58D8      hCPY5-
1486 F58D8 7853         GOSUB  GETDST       Read source info back
```

```
1487                *
1488                * D[A] is destination info, A[W],R0[3:0] is dest. filename,
1489                * B[A] is the file type number, B[S] is the security nibble,
1490                * R2 contains # of nibbles/bytes/records as per file type,
1491                * R3 is record size in bytes
1492                * D1 is destroyed (Points at device info now)
1493                *
1494 F58DC 8E00             GOSUBL =BLANKC
                   00
1495 F58E2 37B6             LCASC  \syek\          Check if keys
                   5697
                   37
1496 F58EC 976              ?A#C   W
1497 F58EF 21               GOYES  hCPY5x          Not keys...OK
1498 F58F1 34C0             LC(5)  =fKEY           Is the type "KEYS"?
                   2E0
1499 F58F8 8A1              ?B=C   A
1500 F58FB 60               GOYES  hCPY5x          Yes...OK
1501                *
1502                * Error...file name is keys, type is NOT keys
1503                *
1504 F58FD 6DEE             GOTO   hCPYt-          Error...Illegal File Type
1505                *-
1506                *-
1507 F5901          hCPY5x
1508                *
1509                * Save R2, R3[A] (R3[15:5]=0), R4[15:5] in FUNCRx RAM
1510                * Save R1[A] (type) in FUNCD0
1511                *
1512 F5901 7793             GOSUB  D0=FR0          Set D0=(5) =FUNCR0
1513 F5905 11A              C=R2
1514 F5908 1547             DATO=C W               Save R2 in FUNCR0
1515 F590C 16F              D0=D0+ 16
1516 F590F 123              AR3EX
1517 F5912 11C              C=R4
1518 F5915 D6               C=A    A               Save R4[15:5], R3[A] in FUNCR1
1519 F5917 113              A=R3                   Restore A[W] (Name)
1520 F591A 1547             DATO=C W               (FUNCR1)
1521 F591E 16F              D0=D0+ 16              (FUNCD0)
1522 F5921 119              C=R1
1523 F5924 144              DATO=C A               Save R1[A] in FUNCD0
1524                *
1525                * Now ready to call FINDF:A[W] is filename, D[S],[B] is device
1526                *
1527 F5927 8F00             GOSBVL =FINDF          Find the file in main RAM
                   000
1528                *
1529                * Now restore R2,R3[A],R4[15:5],R1[A] WITHOUT changing carry
1530                *
1531 F592E 146              C=DAT0 A               (FUNCD0)
1532 F5931 109              R1=C                   Restore R1[A] (type)
1533 F5934 1900             D0=(2) =FUNCR1         (D0=D0- 16 clears carry)
1534 F5938 1567             C=DAT0 W               Read R4[15:5],R3[A] (FUNCR1)
1535 F593C 10C              R4=C                   Restore R4[15:5]
1536 F593F AF0              A=0    W
```

```
1537 F5942 DA          A=C    A            A[W] is now R3 value
1538 F5944 103         R3=A
1539 F5947 1900        D0=(2) =FUNCR0      (D0=D0- 16 clears carry)
1540 F594B 1567        C=DAT0 W
1541 F594F 10A         R2=C                Restore R2[W]
1542             *
1543             * Now check if the file already exists in main RAM
1544             *
1545 F5952 4A0         GOC    hCPY5y       Not found...OK
1546             *
1547             * File exists now...error
1548             *
1549 F5955 7C02        GOSUB  hCPYel       Read any remaining data
1550 F5959 6E34        GOTO   hRNMfx       File exists error
1551             *-
1552             *-
1553 F595D       hCPY5y
1554             *
1555             * Read back the destination info from SAVSTK
1556             *
1557 F595D 73D2        GOSUB  GETDST
1558             *
1559             * Create the destination file now
1560             *
1561             * First save 1 RSTK level in FUNCR0 (D0 now at SCRTCH),
1562             * status bits in FUNCR0+5
1563             *
1564 F5961 07          C=RSTK
1565 F5963 7533        GOSUB  D0=FR0
1566 F5967 144         DAT0=C A            Save stack level in FUNCR0
1567 F596A 164         D0=D0+ 5
1568 F596D 09          C=ST                Save status bits...
1569 F596F 15C2        DAT0=C 3            ...write out status bits
1570 F5973 8F00        GOSBVL =CRTF        Create the file in RAM
          000
1571 F597A 7E13        GOSUB  D0=FR0
1572 F597E 142         A=DAT0 A            Restore stack level from FUNCD1
1573 F5981 DE          ACEX   A            Save error code in A[A]
1574 F5983 06          RSTK=C
1575 F5985 1900        D0=(2) (=FUNCR0)+5  (D0=D0+5 will destroy carry)
1576 F5989 15E2        C=DAT0 3            Read in old status bits...
1577 F598D 0A          ST=C                ...restore status bits
1578 F598F 571         GONC   hCPY5d       No error if no carry
1579             *
1580             * Save the error code in (FUNCR0)+5 for now
1581             *
1582 F5992 1583        DAT0=A 4            Write out 4 nibs of error code
1583             *
1584             * Now clean up the loop (if needed)
1585             *
1586 F5996 7BC1        GOSUB  hCPYel
1587             *
1588             * Recall the error # from (FUNCR0)+5
1589             *
1590 F599A 1B00        D0=(5) (=FUNCR0)+5
```

```
            000
1591 F59A1 15E3         C=DAT0 4
1592 F59A5 02           RTNSC                   Error! (Set up in C[3:0])
1593            *-
1594            *-
1595            *
1596            * Now D[S] is device code, D[X] is device address, R1 is start
1597            * of file header in memory, D1 points to start of data in file
1598            *
1599 F59A7 111  hCPY5d  A=R1
1600 F59AA D2          C=0    A
1601 F59AC 3141        LC(2)  =oFLAGh          Offset to flags...
1602 F59B0 CA          A=A+C  A
1603 F59B2 133         AD1EX                   Save start of data in A[A]
1604            *
1605            * Now D1 points to the flag nibble
1606            *
1607 F59B5 11C         C=R4
1608 F59B8 1554        DAT1=C S                Write out the protection nibble
1609 F59BC 1F00        D1=(5) (=STMTR1)+14     Go to create code
            000
1610 F59C3 1574        C=DAT1 S                Read into C[S]
1611 F59C7 131         D1=A                    Restore start of data
1612            *
1613            * Now get data length back from R2[A] (nibbles)
1614            *
1615 F59CA 112         A=R2
1616            *
1617            * A[A] is now data length in nibbles, C[S] is create code
1618            *
1619 F59CD 80DF        P=C    15
1620 F59D1 D2          C=0    A                Clear high nibbles
1621 F59D3 881         ?P#    1                TITAN data file?
1622 F59D6 90          GOYES  hCPY5,           No...continue
1623 F59D8 20          P=     0
1624 F59DA 308         LC(1)  (=oDAsod)-5      Amount of offset
1625 F59DD EA          A=A-C  A
1626 F59DF 20   hCPY5,  P=     0
1627 F59E1 305         LC(1)  =lFLENh          Length of length field
1628 F59E4 EA          A=A-C  A
1629 F59E6 822         SB=0                    Clear flag for extra nibble
1630 F59E9 25          P=     5
1631 F59EB A80         A=0    P                Clear nibble...
1632 F59EE 81C         ASRB                    ...for bit shift
1633 F59F1 20          P=     0
1634            *
1635            * A[A] is now data length in bytes, SB is 1 if extra nibble
1636            *
1637 F59F3 821         XM=0                    Convert XM to SB value
1638 F59F6 832         ?SB=0
1639 F59F9 60          GOYES  hCPY58
1640 F59FB 7EBA        GOSUB  hCPYXM           Set XM bit
1641 F59FF 102  hCPY58  R2=A                   Save back in R2 for now
1642 F5A02 843         ST=0   =sDEST
1643 F5A05 7052        GOSUB  Rdinfo           Get source info back (addr)
```

```
1644 F5A09 817          DSRC                Rotate address into D[X]
1645 F5A0C 751A         GOSUB  Getmbx       Get mailbox address back
1646 F5A10 870          ?ST=1  =sLoop?      Is this LOOP or non-MS device?
1647 F5A13 B1           GOYES  hCPY5f       Yes...skip SEEKA, DDT
1648 F5A15 114          A=R4
1649 F5A18 8E00         GOSUBL =ASRC5       Get starting address of file
          00
1650 F5A1E 7C02         GOSUB  Seeka        Seek that record
1651 F5A22 473          GOC    hCPYER       Error
1652              *
1653              * Now at the correct record...read the record, check status
1654              *
1655 F5A25 8E00         GOSUBL =DdtRd       Read tape
          00
1656 F5A2B 4E2          GOC    hCPYER
1657              *
1658              * First set D1 to correct location:
1659              *
1660              * Type: 8 - Start of header + oIMPLh + osod (from POLL)
1661              *       4 - Start of header + oIMPLh        (LIF1 file)
1662              *       2 - Start of header + oIMPLh        (41C data file)
1663              *       1 - Start of header + oIMPLh + 8    (TITAN data file)
1664              *       0 - Start of header + oIMPLh        (BASIC, KEYS, etc)
1665              *
1666 F5A2E 111  hCPY5f  A=R1                Start of file header in memory
1667 F5A31 D2           C=0    A
1668 F5A33 3152         LC(2)  =oIMPLh
1669 F5A37 CA           A=A+C  A            Skip first part of header
1670 F5A39 D2           C=0    A
1671 F5A3B 1F00         D1=(5) (=STMTR1)+14 Create code...
          000
1672 F5A42 1574         C=DAT1 S            ...into C[S]
1673 F5A46 94A          ?C=0   S            Mainframe?
1674 F5A49 32           GOYES  hCPY59       Yes...
1675 F5A4B A46          C=C+C  S            Implementation (OEM)?
1676 F5A4E 5E0          GONC   hCPY5&       No...
1677 F5A51 1C8          D1=D1- 9            Point to offset field
1678 F5A54 14F          C=DAT1 B            Read it
1679 F5A57 541          GONC   hCPY59       Go always
1680              *-
1681              *-
1682 F5A5A 4C1  hCPYER  GOC    hCPYE5       Go always...purge the file, error
1683              *-
1684              *-
1685 F5A5D A46  hCPY5&  C=C+C  S            ASCII file?
1686 F5A60 4B0          GOC    hCPY59       Yes...
1687 F5A63 A46          C=C+C  S            41C data file?
1688 F5A66 450          GOC    hCPY59       Yes...
1689              *
1690              * TITAN data file
1691              *
1692 F5A69 308          LC(1)  (=oDAsod)-5  Offset to start of data - link
1693              *
1694 F5A6C CA   hCPY59  A=A+C  A            A[A] points to start of data area
1695 F5A6E 131          D1=A                Point D1 to start of data area
```

```
1696                *
1697                * Set terminate modes to none before copy
1698                *
1699 F5A71 8E00              GOSUBL =CLMODE       Clear terminate modes
           00
1700 F5A77 4A6  hCPYE5      GOC    hCPYEL        Error clearing modes
1701                *
1702                * Now ready to copy the data area of the file
1703                *
1704 F5A7A 112             A=R2                 Read back file length from R2
1705 F5A7D 8A8             ?A=0   A             Is the length zero?
1706 F5A80 11              GOYES  hCPY5z         Yes...don't call READSU (sends SDA)
1707 F5A82 7F81            GOSUB  hCPY5s         Set up send data/set frame count
1708                *
1709 F5A86 D6              C=A    A             ...limit is A[A] bytes
1710 F5A88 8E00            GOSUBL =READSU       Read that many bytes to @ D1
           00
1711 F5A8E 435             GOC    hCPYEL        Error during read
1712 F5A91 831  hCPY5z     ?XM=0                Need 1 more nibble?
1713 F5A94 22              GOYES  hCPY5+         No...continue
1714 F5A96 7B71            GOSUB  hCPY5s         Set up send data/set frame count
1715                *
1716 F5A9A D2              C=0    A
1717 F5A9C E6              C=C+1  A             Read 1 byte to get last nibble
1718 F5A9E DA              A=C    A             Needed for hCPYel (if error)
1719 F5AA0 8E00            GOSUBL =PUTE
           00
1720 F5AA6 4B3             GOC    hCPYEL        Error
1721 F5AA9 8E00            GOSUBL =GETD         Read the data byte (nibble)
           00
1722 F5AAF 423             GOC    hCPYEL        Error
1723 F5AB2 15D0            DAT1=C 1             Write the one nibble out to RAM
1724 F5AB6 860  hCPY5+     ?ST=0  =sLoop?       Is this a mass storage transfer?
1725 F5AB9 22              GOYES  hCPY5i         Yes...go on
1726                *
1727                * For hCPYeL to return, P must be zero!
1728                *
1729 F5ABB 20              P=     0
1730 F5ABD D0              A=0    A             A[A]=0 (have read all bytes)
1731 F5ABF 7450            GOSUB  hCPYeL        No...read the rest of the data
1732 F5AC3 8E00            GOSUBL =GETDev       Am I controller?
           00
1733 F5AC9 4D0             GOC    hCPY5m        No...skip cleanup
1734 F5ACC 96B             ?D=0   B             Is this "LOOP"?
1735 F5ACF 80              GOYES  hCPY5m        Yes...skip cleanup
1736 F5AD1 8E00            GOSUBL =UTLEND       Yes...clean up the loop
           00
1737 F5AD7 6801 hCPY5m     GOTO   hCPY5l        Go check error, etc
1738                *-
1739                *-
1740 F5ADB 7941 hCPY5i     GOSUB  Endtap        Clean up tape business, Loop
1741 F5ADF 57F             GONC   hCPY5m        no error...continue
1742 F5AE2      hCPYEL
1743                *
1744                * Entry to purge mainframe file, then hCPYeL
```

```
1745                    *
1746                    * First save A[A], P, C[0] in R3
1747                    *
1748 F5AE2 816                 CSRC                    C[S] is C[0]
1749 F5AE5 80FE                CPEX    14              C[14] is P
1750 F5AE9 D6                  C=A     A
1751 F5AEB 10B                 R3=C
1752 F5AEE 7241                GOSUB   GETDST          Read destination info
1753                    *
1754                    * Now D[S] is correct for this file, A[W] is filename
1755                    *
1756 F5AF2 119                 C=R1                    Get file header start
1757 F5AF5 135                 D1=C
1758 F5AF8 17F                 D1=D1+  16              Position to file type
1759 F5AFB D2                  C=0     A
1760 F5AFD 15D3                DAT1=C  4               Make sure type is not LEX
1761 F5B01 1CF                 D1=D1-  16              Set D1 back at start of file
1762 F5B04 8F00                GOSBVL  =PRGFMF         Purge the file (partial) file
          000
1763 F5B0B 11B                 C=R3
1764 F5B0E DA                  A=C     A
1765 F5B10 80DE                P=C     14
1766 F5B14 812                 CSLC
1767 F5B17              hCPYeL
1768                    *
1769                    * Entry for P, C[0] = error message, R2[A] is # to have been
1770                    * read, A[A] is number NOT read yet of R2 count, R4[14:10] is
1771                    * number of sectors to be read (total)
1772                    *
1773 F5B17 80C1                C=P     1
1774 F5B1B 8E00                GOSUBL  =TSAV2C         Save error stuff in FUNCR1
          00
1775                    *
1776                    * Set up R4[14:10] to reflect the number of sectors LEFT,
1777                    * A[A] the number of bytes within the current sector, XM=1 if
1778                    * R2[A] is one byte short of real count
1779                    *
1780 F5B21 D8                  B=A     A               Save A[A] in B[A]
1781 F5B23 112                 A=R2                    Get count to A[A]
1782 F5B26 E0                  A=A-B   A               Now A[A] is # actually read
1783 F5B28 831                 ?XM=0
1784 F5B2B 40                  GOYES   hCPYe0          No extra byte
1785 F5B2D E4                  A=A+1   A               Extra byte!
1786 F5B2F D8           hCPYe0  B=A     A               Save count read in B[A]
1787 F5B31 F4                  ASR     A
1788 F5B33 F4                  ASR     A               Now A[A] is # sectors
1789 F5B35 11C                 C=R4
1790 F5B38 7B21                GOSUB   Csrc10
1791 F5B3C E2                  C=C-A   A
1792 F5B3E 431                 GOC     hCPYex
1793 F5B41 D0                  A=0     A
1794 F5B43 B60                 A=A-B   B               Now A[A] is # bytes to read
1795 F5B46 8AC                 ?A#0    A               Is it non-zero?
1796 F5B49 50                  GOYES   hCPYe+          Yes...OK as is
1797 F5B4B B24                 A=A+1   XS              No...full sector
```

```
1798 F5B4E 7320 hCPYe+  GOSUB  hCPYe.        Read them
1799 F5B52 8E00 hCPYex  GOSUBL =TRES2C       Restore the error stuff
          00
1800 F5B58 80D1         P=C    1
1801 F5B5C 890          ?P=    0             If P=0, return (not error)
1802 F5B5F 00           RTNYES
1803 F5B61 62CA hCPYeR  GOTO   hCPYer
1804            *-
1805            *-
1806 F5B65 7CB8 hCPYe1  GOSUB  Getmbx        Set D0 back to the mailbox
1807 F5B69 D0   hCPYe-  A=0    A
1808 F5B6B B24         A=A+1  XS            Set A[A]=#100 (256)
1809 F5B6E 11C         C=R4
1810 F5B71 72F0         GOSUB  Csrc10        Get # of sectors into C[A]
1811            *
1812            * Check if not loop or non-MS device...if so, return
1813            *
1814 F5B75 860  hCPYe.  ?ST=0  =sLoop?
1815 F5B78 44          GOYES  hCPYe4         Set P=0, return
1816 F5B7A CE          C=C-1  A              Decrement by 1
1817 F5B7C 7A50        GOSUB  Cslc10         Put it back
1818 F5B80 10C         R4=C
1819 F5B83 483         GOC    hCPYe4         If carry, done with reads
1820 F5B86 7B98        GOSUB  Getmbx         Get the mailbox back
1821 F5B8A 7780 hCPYe1  GOSUB  hCPY5s        Set up send data/set frame count
1822            *
1823 F5B8E D6          C=A    A              Get count into C[A] (frame count)
1824 F5B90 8E00        GOSUBL =PUTE          Send it to start conversation
          00
1825 F5B96 452         GOC    hCPYe4         Error if carry
1826 F5B99 8A8  hCPYe2  ?A=0   A
1827 F5B9C DC          GOYES  hCPYe-
1828 F5B9E 8E00        GOSUBL =GETX          Read the data
          00
1829 F5BA4 582         GONC   hCPYe3         Got a data byte...process it
1830 F5BA7 880         ?P#    0              Is this a EOT?
1831 F5BAA 21          GOYES  hCPYe4         Definitely not...error!
1832 F5BAC 8E00        GOSUBL =FRAME-        Check for EOT
          00
1833 F5BB2 890         ?P=    =pTERM         Is it terminator match? (possible)
1834 F5BB5 5D          GOYES  hCPYe1         Yes...restart it
1835 F5BB7 890         ?P=    =pEOT          Is it specifically EOT?
1836 F5BBA 0D          GOYES  hCPYe1         Yes...restart it
1837 F5BBC 20   hCPYe4  P=     0              Common exit code
1838 F5BBE 8E00        GOSUBL =GETDev        Check if device or controller
          00
1839 F5BC4 500         RTNNC                 If controller:return, carry clear
1840 F5BC7 8C00        GOLONG =TER/LF        Terminate on LF/end frame
          00
1841            *-
1842            *-
1843 F5BCD CC   hCPYe3  A=A-1  A
1844 F5BCF 4CE         GOC    hCPYe4         Error (too many)
1845 F5BD2 0D          P=P-1                 Decrement # of bytes
1846 F5BD4 58F         GONC   hCPYe3         More yet
```

```
1847 F5BD7 41C          GOC    hCPYe2        Done with this one...go on
1848            *-
1849            *-
1850 F5BDA 8C00 =Cslc10 GOLONG =CSLC10
          00
1851            *-
1852            *-
1853            *
1854            * Check if this is a lex file...if so, add it to LEX tables
1855            *
1856 F5BE0 111  hCPY51  A=R1                  Get back start of file
1857 F5BE3 102          R2=A                  Save in R2, in case call LEXBF+
1858 F5BE6 20           P=     0
1859 F5BE8 D2           C=0    A             Clear the high nibbles first
1860 F5BEA 3101         LC(2)  =oFTYPh       Offset of TYPE in header
1861 F5BEE CA           A=A+C  A
1862 F5BF0 131          D1=A
1863 F5BF3 D0           A=0    A             Clear high nibble
1864 F5BF5 15B3         A=DAT1 4
1865 F5BF9 3380         LC(4)  =fLEX         LEX file type
          2E
1866 F5BFF 8A6          ?A#C   A             Is this LEX?
1867 F5C02 F0           GOYES  hCPY5e        No...exit
1868 F5C04 8F00         GOSBVL =LEXBF+       Yes...update the LEX buffers
          000
1869 F5C0B 11A          C=R2
1870 F5C0E 109          R1=C                 Restore start of file from R2
1871 F5C11 6D6B hCPY5e  GOTO   RtnXM0        Clear XM for sure to finish
1872            *-
1873            *-
1874 F5C15 25   =hCPY5s P=     5
1875 F5C17 300          LC(1)  =mSDA@5       Assume controller mode...
1876 F5C1A 8E00         GOSUBL =GETDev       Sets carry if device
          00
1877 F5C20 500          RTNNC                (controller...done)
1878 F5C23 300          LC(1)  =mSFC@5       Device mode...set frame count
1879 F5C26 03           RTNCC                Force carry clear
1880            *-
1881            *-
1882 F5C28 8C00 =Endtap GOLONG =ENDTAP
          00
1883            *-
1884            *-
1885 F5C2E 8C00 =Seeka  GOLONG =SEEKA
          00
1886            *-
1887            *-
1888 F5C34 7E10 GETDST  GOSUB  Rdinfd        Get destination information first
1889 F5C38 1B00         D0=(5) =SCRTCH
          000
1890 F5C3F 97C          ?A#0   W             Filename defined?
1891 F5C42 60           GOYES  GETDS1        Yes...check device type
1892 F5C44 1527         A=DAT0 W             No...read source name
1893 F5C48 817  GETDS1  DSRC                 Rotate device into D[S]
1894 F5C4B B47          D=D+1  S             Check if device is specified...
```

```
1895 F5C4E 400          RTNC                ...no...return with mainframe
1896 F5C51 A4F          D=D-1  S            Specified...restore it
1897 F5C54 03           RTNCC
1898            *-
1899            *-
1900 F5C56 853  Rdinfd  ST=1   =sDEST
1901 F5C59 8C00 Rdinfo  GOLONG =RDINFO
          00
1902            *-
1903            *-
1904 F5C5F D2   Gt2zer  C=0    A            Clear high nibs of C before call
1905 F5C61 8C00 Gt2byt  GOLONG =GT2BYT
          00
1906            *-
1907            *-
1908 F5C67 8C00 Csrc10  GOLONG =CSRC10
          00
1909            *-
1910            *-
1911 F5C6D 8C00 =Findf+ GOLONG =FINDF+
          00
1912            *-
1913            *-
1914 F5C73 20   =DdlPwr P=     =PWrite
1915 F5C75 7410         GOSUB  Ddl
1916 F5C79 400          RTNC
1917 F5C7C 8E00         GOSUBL =TSTAT
          00
1918 F5C82 400          RTNC
1919 F5C85 8C00 Mtyl    GOLONG =MTYL
          00
1920            *-
1921            *-
1922 F5C8B 20   DdlWrt  P=     =Write
1923 F5C8D 8C00 Ddl     GOLONG =DDL
          00
1924            *-
1925            *-
1926 F5C93 1F00 =D1=S20 D1=(5) (=SCRTCH)+20
          000
1927 F5C9A 01           RTN
1928            *-
1929            *-
1930 F5C9C 1B00 =D0=FR0 D0=(5) =FUNCR0
          000
1931 F5CA3 01           RTN
1932            ****************************************************************
1933            ****************************************************************
1934            **
1935            ** Name:       hPURGE - PURGE statement POLL handler (HPIL)
1936            **
1937            ** Category:   POLL
1938            **
1939            ** Purpose:
1940            **      Handle the PURGE statement POLL if HPIL device
```

```
1941                **
1942                ** Entry:
1943                **      Name in A[W], R0[3:0]
1944                **      Device in D[S], D[X]
1945                **      P=0,HEXMODE
1946                **      Destination info on SAVSTK (under POLLSV)
1947                **
1948                ** Exit:
1949                **      P=0
1950                **      Carry set: Error (C[3:0] is error number)
1951                **      Carry clear:
1952                **        XM=0: handled...FIB file start zeroed, file purged
1953                **              ST[8]=0 (Current file not purged)
1954                **        XM=1: not handled (not HPIL/not Filbert)
1955                **      SAVSTK unchanged from entry
1956                **
1957                ** Calls:      CKBITL,FINDF+,DATST+,SAVDIR,CHKSEC,fPROT,D1=S20,
1958                **              hPUTDR,ENDTAP,I/OFND
1959                **
1960                ** Uses.......
1961                **  Inclusive: A-D,R0-R3,D0,D1,P,ST[8,5:0],SCRTCH
1962                **
1963                ** Stk lvls:   6 (FINDF+)
1964                **
1965                ** History:
1966                **
1967                **    Date      Programmer              Modification
1968                **  --------    ----------    --------------------------------
1969                **  01/12/83      NZ          Updated documentation
1970                **
1971                ********************************************************************
1972                ********************************************************************
1973 F5CA5 D9      SAVDIR  C=B     A               Save directory pointer in R3
1974 F5CA7 10B             R3=C
1975 F5CAA 71BF    GETTYP  GOSUB   Gt2zer          Read the file type
1976                *
1977                * Now C[A] is the file type...check security!
1978                *
1979 F5CAE DA              A=C     A
1980 F5CB0 8D00    =fTYPF# GOVLNG  =FTYPF#
           000
1981                *-
1982                *-
1983 F5CB7         hPURER
1984 F5CB7 8C00    =Error  GOLONG  =ERROR          Set up error, return w/carry set
           00
1985                *-
1986                *-
1987 F5CBD         =hPURGE
1988 F5CBD 73CA            GOSUB   CKBITL
1989 F5CC1 500             RTNNC                   If no carry, not (HPIL&Filbert)
1990                *
1991                * This IS an HPIL purge!
1992                *
1993                * Save filename in R0, R1, START,CHKMAS,FINDFx
```

```
1994            *
1995 F5CC4 75AF         GOSUB  Findf+
1996            *
1997            * If file not found, carry will be set...Error, not warning!
1998            *
1999 F5CC8 4EE          GOC    hPURER
2000            *
2001            * Save file information in R2 (to clean up FIB)
2002            * R2[6:4] is device address, R2[3:0] is data start address
2003            *
2004 F5CCB 8E2D         GOSUBL DATST+
           4F
2005 F5CD1 10A          R2=C                    Save it in R2
2006            *
2007            * Save the directory information in R1 now
2008            *
2009 F5CD4 7DCF         GOSUB  SAVDIR           Save dir pointer in R3, get type
2010 F5CD8 573          GONC   hPUR20           If no carry, didn't find type
2011            *
2012            * Found it...check if secure (if so, error...can't purge it)
2013            *
2014 F5CDB 7C00         GOSUB  CHKSEC           Check if secure
2015 F5CDF 503          GONC   hPUR20           Not secure...ok to purge
2016            *
2017            * This is a secure file...can't purge it
2018            *
2019 F5CE2 8E00 hPURSC  GOSUBL =fPROT           Protected file error (P, C[0])
           00
2020 F5CE8 4EC          GOC    hPURER           Go always (set up error, RTNSC)
2021            *-
2022            *-
2023 F5CEB A4D  =CHKSEC B=B-1  S                Convert to base zero
2024 F5CEE AC9          C=B    S
2025 F5CF1 80DF         P=C    15
2026 F5CF5 891          ?P=    1
2027 F5CF8 00           RTNYES                  Secure
2028 F5CFA 893          ?P=    3
2029 F5CFD 00           RTNYES                  Secure, private
2030 F5CFF 03           RTNCC
2031            *-
2032            *-
2033 F5D01 11B  hPUTDR  C=R3
2034 F5D04 816          CSRC
2035 F5D07 AD2          C=0    M                Clear all unneeded nibbles
2036 F5D0A 8C00         GOLONG =PUTDR#          Write the entry from SCRTCH
           00
2037            *-
2038            *-
2039 F5D10      hPUR20
2040            *
2041            * OK to purge it
2042            *
2043 F5D10 7F7F         GOSUB  D1=S20           Set D1= (=SCRTCH)+20
2044 F5D14 D2           C=0    A
2045 F5D16 15D3         DAT1=C 4                Set file type = 0
```

```
2046                *
2047                * Now record # in C[A], directory entry # in C[S]
2048                *
2049 F5D1A 73EF       GOSUB  hPUTDR        Write the entry from SCRTCH
2050 F5D1E 489        GOC    hPURER        Error during write
2051                *
2052                * Now clean up the tape, etc
2053                *
2054 F5D21 730F       GOSUB  Endtap        Clean up tape (rewind, etc)
2055 F5D25 419        GOC    hPURER        Error during clean-up
2056 F5D28 848        ST=0   8             Current file was not purged
2057 F5D2B 3230       LC(3)  =bFIB
           8
2058 F5D30 8E00       GOSUBL =1/OFND
           00
2059 F5D36 11A        C=R2
2060                *
2061                * Entry to purge an FIB entry (D1 @ FIB buffer, C is pointer)
2062                *
2063 F5D39 26    =PURFIB P=     6
2064 F5D3B 14B    FNDENT A=DAT1 B
2065 F5D3E 968        ?A=0   B
2066 F5D41 72         GOYES  NOTFND
2067 F5D43 17C        D1=D1+ =oFBEGb
2068 F5D46 177        D1=D1+ (oDBEGb)-(oFBEGb)
2069 F5D49 15B6       A=DAT1 7
2070 F5D4D 912        ?A=C   WP
2071 F5D50 E0         GOYES  FIXIT
2072 F5D52 17E        D1=D1+ (oRECLb)-(oDBEGb)
2073 F5D55 17F        D1=D1+ (oRLENb)-(oRECLb)
2074 F5D58 17A        D1=D1+ (1FIB)-(oRLENb)
2075 F5D5B 5FD        GONC   FNDENT
2076                *
2077 F5D5E 1C7    FIXIT  D1=D1- (oDBEGb)-(oFBEGb)
2078 F5D61 AF2        C=0    W
2079 F5D64 15D5       DAT1=C 6
2080                *
2081 F5D68 20     NOTFND P=     0
2082 F5D6A 641A       GOTO   RtnXMO
2083                ****************************************************************
2084                ****************************************************************
2085                **
2086                ** Name:       hRENAM - HPIL handler for the RENAME POLL
2087                **
2088                ** Category:   POLL
2089                **
2090                ** Purpose:
2091                **      HPIL handler for RENAME execute POLL
2092                **
2093                **
2094                ** Entry:
2095                **      A[W] is first 8 chars of filename
2096                **      R0[3:0] is last 2 chars
2097                **      D[3:0],D[S] is source device information
2098                **      P=0
```

```
2099                **      Source, destination info on SAVSTK (under POLLSV)
2100                **
2101                ** Exit:
2102                **      P=0
2103                **      Carry set: Error...error # in C[3:0]
2104                **      Carry clear:
2105                **        XM=0: handled
2106                **        XM=1: not handled
2107                **
2108                ** Calls:      CKBITL,hRNMsb,FINDF+,FINDFx,SAVDIR,D1=SCR,hPUTDR,
2109                **             ENDTAP
2110                **
2111                ** hRNMsb calls RDINFO
2112                **
2113                ** Uses.......
2114                **  Inclusive: A-D,R0,R1,R3,D0,D1,P,ST[8,5:0],SCRTCH
2115                **
2116                ** Stk lvls:   6 (FINDF+)
2117                **
2118                ** History:
2119                **
2120                **    Date      Programmer          Modification
2121                **  --------    ----------    ------------------------------
2122                **  06/02/83       NZ         Rewrote parts to pack code and
2123                **                            share routines with PURGE, SECURE
2124                **  01/13/83       NZ         Fixed bug in hRNMsb (setup for
2125                **                              FINDFx was incorrect)
2126                **                            Changed very first part of hRENAM
2127                **  01/12/83       NZ         Updated documentation
2128                **
2129                ********************************************************************
2130                ********************************************************************
2131 F5D6E 721A =hRENAM GOSUB  CKBITL
2132 F5D72 500         RTNNC                 Not HPIL filbert...returnCC, XM=1
2133                *
2134                * Source or destination is HPIL (D[A] is address)
2135                *
2136                * A[W] is first 8 chars of source name, R0[3:0] is last 2 char
2137                * D[X] is HPIL address, D[S] is "8"
2138                *
2139 F5D75 7070          GOSUB  hRNMsd
2140 F5D79 70FE          GOSUB  Findf+         Find the destination file
2141                *
2142                * If found, error (File exists already)
2143                *
2144 F5D7D 5A1           GONC   hRNMfx         Error...file exists already
2145                *
2146                * Check if error is "file not found" or something else
2147                *
2148 F5D80 880           ?P#    =eTAPE         Is it tape error?
2149 F5D83 A1            GOYES  hRNMER         No..."real" error
2150 F5D85 80F0          CPEX   0
2151 F5D89 880           ?P#    =eNFILE        Is it "No file" (Not found)?
2152 F5D8C 20            GOYES  hRNM25
2153 F5D8E 80F0 hRNM25   CPEX   0             (Carry clear=not found)
```

```
2154 F5D92 521          GONC   hRNM30        Not found...continue
2155 F5D95 470          GOC    hRNMER        Go always - tape error
2156            *-
2157            *-
2158 F5D98 300  hRNMfx  LC(1)  =eEFILE       File already exists
2159 F5D9B 20           P=     =eTAPE
2160 F5D9D 691F hRNMER  GOTO   Error         Set up error code, RTNSC
2161            *-
2162            *-
2163 F5DA1 6BF9 hRNMXM  GOTO   hCPYxm        Carry clear, XM=1
2164            *-
2165            *-
2166            *
2167            * Destination file not found...continue
2168            *
2169 F5DA5 843  hRNM30  ST=0   =sDEST
2170 F5DA8 7040         GOSUB  hRNMsb
2171 F5DAC 120          AROEX                 Put first 8 chars in R0...
2172 F5DAF 101          R1=A                  ...and last 2 chars in R1
2173 F5DB2 8E00         GOSUBL =FINDFx        Find the source file
          00
2174 F5DB8 44E          GOC    hRNMER        Error
2175            *
2176            * Now the B[3:0] is the directory pointer for the file
2177            *
2178 F5DBB 76EE         GOSUB  SAVDIR         Save directory info, get file type
2179            *                             (Ignore carry from FTYPF#)
2180            *
2181            * Now get the destination name back
2182            *
2183 F5DBF 7620         GOSUB  hRNMsd         Get back the destination info
2184            *
2185            * Now A[W] is the first 8 chars, R0 is the last 2 chars
2186            *
2187 F5DC3 8E00         GOSUBL =D1=SCR        Point D1 @ SCRTCH
          00
2188 F5DC9 1517         DAT1=A W              Write out first 8 chars of name
2189 F5DCD 17F          D1=D1+ 16             Position to last 2 chars location
2190 F5DD0 110          A=R0
2191 F5DD3 1593         DAT1=A 4              Write out last 2 chars of name
2192 F5DD7 762F         GOSUB  hPUTDR         Write directory entry from SCRTCH
2193 F5DDB 41C          GOC    hRNMER         Error
2194 F5DDE 764E         GOSUB  Endtap         End the tape conversation
2195 F5DE2 4AB          GOC    hRNMER         Error
2196 F5DE5 6999         GOTO   RtnXMO         Return, indicate "handled"
2197            *-
2198            *-
2199 F5DE9 853  hRNMsd  ST=1   =sDEST         Set destination first
2200 F5DEC DB   hRNMsb  C=D    A
2201 F5DEE 109          R1=C                  Save address in R1[A]
2202 F5DF1 746E         GOSUB  Rdinfo
2203 F5DF5 AFB          C=D    W              Save dest device & address in R1
2204 F5DF8 129          CR1EX                 Restore old address, save new
2205 F5DFB D7           D=C    A              Restore address to D[A]
2206 F5DFD 03           RTNCC                 Carry clear
```

```
2207            ****************************************************************
2208            ****************************************************************
2209            **
2210            ** Name:       hFPROT - File protection handler (HPIL files)
2211            **
2212            ** Category:   POLL
2213            **
2214            ** Purpose:
2215            **      Execute the SECURE/PRIVATE command for an HPIL device
2216            **
2217            ** Entry:
2218            **      D[S] is the device type: if HPIL, then A[W] is first
2219            **      8 chars of filename, R0[3:0] is last 2 chars, D[X] is
2220            **      HPIL address of the device
2221            **      Destination info on SAVSTK (under POLLSV)
2222            **      (See detail also!)
2223            **
2224            ** Exit:
2225            **      Carry set: Error (C[3:0] is error number)
2226            **      Carry clear:
2227            **        XM=1: Not handled (not HPIL/not Filbert)
2228            **        XM=0: Handled (action taken)
2229            **
2230            ** Calls:      CKBITL,FINDF+,SAVDIR,CHKSEC,D1=S20,PT2BYT,
2231            **             hPUTDR,ENDTAP
2232            **
2233            ** Uses.......
2234            **  Inclusive: A-D,R0,R1,R3,D0,D1,P,ST[8,5:0],SCRTCH
2235            **
2236            ** Stk lvls:   6 (FINDF+)
2237            **
2238            ** Detail:
2239            **      ST(sPRIVT) set if PRIVATE, clear if SECURE
2240            **      ST(sUNSEC) set if UNSECURE, clear if SECURE
2241            **
2242            ** History:
2243            **
2244            **    Date      Programmer            Modification
2245            **  --------    ----------    --------------------------------
2246            **  06/02/83      NZ          Reworked to share much code with
2247            **                            PURGE and RENAME
2248            **  02/08/83      NZ          Changed to prevent PRIVATE on a
2249            **                              secure file (design change)
2250            **  01/12/83      NZ          Converted to single poll entry
2251            **  12/20/82      NZ          Added routine and documentation
2252            **
2253            ****************************************************************
2254            ****************************************************************
2255 F5DFF 6D80 hSECeR  GOTO   hSECer         Error jump
2256            *-
2257            *-
2258 F5E03 7D79 =hFPROT GOSUB  CKBITL         Check if this is HPIL & Filbert
2259 F5E07 500          RTNNC                No...set XM (not handled)
2260            *
2261            * This is an HPIL device
```

```
2262            *
2263 F5E0A 7F5E         GOSUB  Findf+        Save A in R0, R0>R1, START,FINDFx
2264 F5E0E 40F          GOC    hSECeR        Error
2265            *
2266            * Have found the file (D1 is at file type)
2267            *
2268 F5E11 709E         GOSUB  SAVDIR        Save dir info in R3, check type
2269 F5E15 460          GOC    hSEC15        Found type entry...continue
2270 F5E18 66D9 hSECft  GOTO   hCPYtp        Not found...error
2271            *-
2272            *-
2273            *
2274            * Found it...C[A], B[A] point to the entry, B[S] is position
2275            * of the type within the entry
2276            *
2277 F5E1C 7BCE hSEC15  GOSUB  CHKSEC        Check if secure(leaves P=entry #)
2278 F5E20 0B           CSTEX
2279 F5E22 80F0         CPEX   0
2280 F5E26 0B           CSTEX                Now ST[3:0] is the current pos
2281            sSEC    EQU    0             Bit for SECURE
2282            sPR     EQU    1             Bit for PRIVATE
2283 F5E28 860          ?ST=0  =sPRIVT       Is this PRIVATE statement?
2284 F5E2B E0           GOYES  hSEC20        No...must be secure
2285            *
2286            * PRIVATE statement
2287            *
2288 F5E2D 851          ST=1   sPR           Make it private!
2289 F5E30 860          ?ST=0  sSEC          Is it OK (NOT secure)?
2290 F5E33 41           GOYES  hSEC30        Yes...write it back out
2291 F5E35 6CAE         GOTO   hPURSC        No...file secure
2292            *-
2293            *-
2294 F5E39      hSEC20
2295            *
2296            * [UN]SECURE statement (need to determine which it is)
2297            *
2298 F5E39 860          ?ST=0  =sUNSEC       UNSECURE?
2299 F5E3C 80           GOYES  hSEC25        No...must be SECURE statement
2300            *
2301            * This is the UNSECURE statement
2302            *
2303 F5E3E 840          ST=0   sSEC          Clear the security bit
2304 F5E41 550          GONC   hSEC30        Go always
2305            *-
2306            *-
2307 F5E44      hSEC25
2308            *
2309            * This is the SECURE statement
2310            *
2311 F5E44 850          ST=1   sSEC
2312 F5E47      hSEC30
2313            *
2314            * Now ST[3:0] is the desired entry #
2315            *
2316 F5E47 0B           CSTEX
```

```
2317 F5E49 80F0          CPEX   0            Restore ST[3:0] from P
2318 F5E4D 0B            CSTEX
2319 F5E4F 80CF          C=P    15           Set C[S] to desired security
2320              *
2321              * Now C[S] is the desired type #, C[A] is the entry address
2322              *
2323 F5E53 135           D1=C
2324 F5E56 17E           D1=D1+ 15           Point to # types
2325 F5E59 1534          A=DAT1 S            Read it in...
2326 F5E5D 9CA           ?A<=C  S            ...is the type I want available?
2327 F5E60 8B            GOYES  hSECft       No...file type error
2328 F5E62 1C4           D1=D1- 5            Position to (type-2)
2329 F5E65 173  hSEC40   D1=D1+ 4            Go to next type
2330 F5E68 A4E           C=C-1  S            Done yet?
2331 F5E6B 59F           GONC   hSEC40       No...loop back
2332              *
2333              * Now D1 is at the desired file type
2334              *
2335 F5E6E 15F5          C=DAT1 6            Read type into C[5:2]
2336 F5E72 7D1E          GOSUB  D1=S20       Point to the type
2337 F5E76 8E00          GOSUBL =PT2BYT      Write the new file type
           00
2338              *
2339              * Now get the pointer back from R3 and write the entry
2340              *
2341 F5E7C 718E          GOSUB  hPUTDR       Write the entry from SCRTCH
2342 F5E80 4C0           GOC    hSECer       Error
2343 F5E83 71AD          GOSUB  Endtap       Clean up the loop
2344 F5E87 821           XM=0                Make sure XM=0 (handled)
2345 F5E8A 500           RTNNC               Return if no carry...done
2346              *
2347              * If fall through RTNNC, then error has occurred during ENDTAP
2348              *
2349 F5E8D 692E hSECer   GOTO   Error         Return, carry set
2350 F5E91               END
```

```
 A-MULT  Ext                    -   353
 ACES=0  Abs 1004290 #F5302 -   454   444   567
 ASLC12  Ext                    -  1278
 ASLC3   Ext                    -   334
 ASLC4   Ext                    -  1173
 ASLC6   Ext                    -  1193
 ASRC10  Ext                    -  1204
 ASRC3   Ext                    -  1040
 ASRC4   Ext                    -   995
 ASRC5   Ext                    -  1012  1649
 BLANKC  Ext                    -  1494
 Basic   Abs       0 #00000 -  1182  1183  1188  1199
 CHAIN-  Ext                    -  1214
 CHKASN  Ext                    -   660
 CHKBIT  Ext                    -  1055  1292
 CHKMAS  Ext                    -   259
=CHKSEC  Abs 1006827 #F5CEB -  2023  2014  2277
=CKBITL  Abs 1005444 #F5784 -  1292   115  1988  2131  2258
=CKHPI+  Abs 1005456 #F5790 -  1295   633
=CKHPIL  Abs 1005453 #F578D -  1294   253
 CKHPIx  Abs 1005467 #F579B -  1299  1293  1296
 CLMODE  Ext                    -  1699
 CRTF    Ext                    -  1570
 CRTF00  Abs 1003983 #F51CF -   261   258
 CRTF01  Abs 1003987 #F51D3 -   264   260
 CRTF05  Abs 1004004 #F51E4 -   272   267
 CRTF10  Abs 1004053 #F5215 -   300   286
 CRTF20  Abs 1004074 #F522A -   312   303
 CRTF30  Abs 1004114 #F5252 -   342   313
 CRTF35  Abs 1004134 #F5266 -   348   346
 CRTF4.  Abs 1004049 #F5211 -   297   301
 CRTF40  Abs 1004154 #F527A -   359   297   309   336
 CSLC10  Ext                    -  1850
 CSLC2   Ext                    -  1247
 CSLC3   Ext                    -   755
 CSLC4   Ext                    -   855
 CSLC6   Ext                    -   362
 CSLC7   Ext                    -   182
 CSLC9   Ext                    -   439
 CSRC10  Ext                    -  1908
 CSRC3   Ext                    -   763
 CSRC4   Ext                    -   343   371
 CSRC5   Ext                    -   765   936  1244
 CSRC9   Ext                    -   638
=Cslc10  Abs 1006554 #F5BDA -  1850   938  1817
 Cslc4   Abs 1004722 #F54B2 -   855   184   351
 Cslc5   Abs 1004719 #F54AF -   854   127   753  1013  1319  1321
 Csrc10  Abs 1006695 #F5C67 -  1908  1037  1472  1790  1810
 D0=FIB  Ext                    -   688
=D0=FR0  Abs 1006748 #F5C9C -  1930  1010  1034  1512  1565  1571
=D1=S20  Abs 1006739 #F5C93 -  1926   973  1000  2043  2336
 D1=SCR  Ext                    -  2187
 DATST+  Abs 1003939 #F51A3 -   183  2004
 DATSTR  Abs 1003921 #F5191 -   179   134
 DDL     Ext                    -  1923
```

```
 Ddl     Abs 1006733 #F5C8D -  1923  1915
=DdlPwr  Abs 1006707 #F5C73 -  1914
 DdlWrt  Abs 1006731 #F5C8B -  1922
 DdtRd   Ext                -  1655
 ENDTAP  Ext                -  1882
 ERROR   Ext                -  1984
 ERRORX  Ext                -   448
 ERror   Abs 1004000 #F51E0 -   269   138   256
=Endtap  Abs 1006632 #F5C28 -  1882   136  1059  1740  2054  2194  2343
=Error   Abs 1006775 #F5CB7 -  1984  1068  2160  2349
 Errorx  Abs 1004284 #F52FC -   448   426   432   443   502   552   559   566
                            -   654
 FINDF   Ext                -  1082  1527
 FINDF+  Ext                -  1911
 FINDFL  Ext                -   928  1313
 FINDFx  Ext                -   119  2173
 FIXIT   Abs 1006942 #F5D5E -  2077  2071
 FNDENT  Abs 1006907 #F5D3B -  2064  2075
 FNDMB+  Ext                -  1057
 FRAME-  Ext                -  1832
 FTYPF#  Ext                -  1980
 FUNCRO  Ext                -  1539  1575  1590  1930
 FUNCR1  Ext                -  1533
=Findf+  Abs 1006701 #F5C6D -  1911  1995  2140  2263
 GETBYT  Ext                -  1404
 GETD    Ext                -  1721
 GETDS1  Abs 1006664 #F5C48 -  1893  1891
 GETDST  Abs 1006644 #F5C34 -  1888  1486  1557  1752
 GETDev  Ext                -  1732  1838  1876
 GETMBX  Ext                -   682
 GETTYP  Abs 1006762 #F5CAA -  1975   947  1348
 GETX    Ext                -  1828
 GT2BYT  Ext                -  1905
=Getmbx  Abs 1004581 #F5425 -   682   563   583  1645  1806  1820
 Gt2byt  Abs 1006689 #F5C61 -  1905   181
 Gt2zer  Abs 1006687 #F5C5F -  1904  1975
 LEXBF+  Ext                -  1868
 MOVEFL  Ext                -  1048
 MTYL    Ext                -  1919
 Mtyl    Abs 1006725 #F5C85 -  1919
 NEWFI+  Ext                -  1027  1286
 NEWFIL  Ext                -   377
 NOTFND  Abs 1006952 #F5D68 -  2081  2066
 OUTPTt  Ext                -   741
 PILVER  Ext                -    63
 PLOTt   Ext                -   743
 PRGFMF  Ext                -  1762
 PRTIS+  Ext                -   759
 PT2BYT  Ext                -  2337
=PURFIB  Abs 1006905 #F5D39 -  2063
 PUTDR#  Ext                -  2036
 PUTE    Ext                -  1719  1824
 PWrite  Ext                -  1914
 RDINFO  Ext                -  1901
 RDNBIO  Abs 1004371 #F5353 -   561   556
```

```
 READR#  Ext                  -   431   565
 READSU  Ext                  -  1710
 RSTOR+  Abs 1004252 #F52DC   -   432   505
 RSTORE  Abs 1004255 #F52DF   -   438   584
 Rdinfd  Abs 1006678 #F5C56   -  1900  1272  1888
 Rdinfo  Abs 1006681 #F5C59   -  1901  1643  2202
 RtnXMO  Abs 1005439 #F577F   -  1288   458  1065  1871  2082  2196
 SAVDIR  Abs 1006757 #F5CA5   -  1973  2009  2178  2268
 SCRTCH  Ext                  -  1018  1382  1402  1889  1926
 SEEKA   Ext                  -  1885
 SNAPRS  Ext                  -   445
 START   Ext                  -   685
 STBUF+  Abs 1004436 #F5394   -   630   422   498   548
 STMTR0  Ext                  -   780
 STMTR1  Ext                  -   273   750   778  1609  1671
 STUP10  Abs 1004513 #F53E1   -   657   651
 STUP20  Abs 1004531 #F53F3   -   663   681
 STUPBF  Abs 1004471 #F53B7   -   642   562
=Seeka   Abs 1006638 #F5C2E   -  1885  1650
 Start   Abs 1004587 #F542B   -   685   117   255   425   501   551
 TER/LF  Ext                  -  1840
 TRES2C  Ext                  -   761  1032  1799
 TSAV2C  Ext                  -   757  1022  1774
 TSAVD1  Ext                  -   748
 TSTAT   Ext                  -  1917
 UTLEND  Ext                  -  1736
 Utlend  Ext                  -   442
 WRITE#  Ext                  -   504   558
 WRTADR  Abs 1004561 #F5411   -   676   427   503   553
 Write   Ext                  -  1922
 bFIB    Abs    2051 #00803   -    13  2057
 d0=FIB  Abs 1004593 #F5431   -   688   454   630   642   676
 eEFILE  Ext                  -  2158
 eFTYPE  Ext                  -  1357
 eFnFND  Ext                  -  1083
 eNFILE  Ext                  -  1336  2151
 ePIL    Ext                  -   653
 eRANGE  Ext                  -   268
 eSYSer  Ext                  -   652
 eTAPE   Ext                  -  1337  1343  2148  2159
 eTSIZE  Ext                  -  1342
 fBASIC  Abs   57876 #0E214   -    13  1184
 fKEY    Abs   57868 #0E20C   -    13  1498
 fLEX    Abs   57864 #0E208   -    13  1865
 fPROT   Ext                  -  2019
=fTYPF#  Abs 1006768 #F5CB0   -  1980
=hCOPY%  Abs 1004728 #F54B8   -   858
 hCPY10  Abs 1004739 #F54C3   -   872   860
 hCPY12  Abs 1004801 #F5501   -   908   893
 hCPY22  Abs 1004826 #F551A   -   936   929
 hCPY23  Abs 1004854 #F5536   -   952   948
 hCPY24  Abs 1004876 #F554C   -   971   962
 hCPY25  Abs 1004911 #F556F   -   994   988
 hCPY28  Abs 1005053 #F55FD   -  1054  1064
 hCPY29  Abs 1005080 #F5618   -  1061  1056
```

```
 hCPY3+  Abs 1005218 #F56A2 -  1170  1251  1268
 hCPY3-  Abs 1005212 #F569C -  1165  1228
 hCPY3.  Abs 1004786 #F54F2 -   897   876   879
 hCPY30  Abs 1005096 #F5628 -  1071   905
 hCPY31  Abs 1005143 #F5657 -  1108  1101
 hCPY32  Abs 1005205 #F5695 -  1148  1131
 hCPY33  Abs 1005209 #F5699 -  1155  1129
 hCPY34  Abs 1005321 #F5709 -  1220  1134
 hCPY36  Abs 1005331 #F5713 -  1231  1139
 hCPY37  Abs 1005348 #F5724 -  1244  1240
 hCPY38  Abs 1005377 #F5741 -  1254  1145
 hCPY39  Abs 1005408 #F5760 -  1271  1217
 hCPY3a  Abs 1005191 #F5687 -  1137  1133
 hCPY3b  Abs 1005201 #F5691 -  1145  1138
 hCPY3f  Abs 1005258 #F56CA -  1189  1187
 hCPY3g  Abs 1005317 #F5705 -  1217  1200
 hCPY5!  Abs 1005529 #F57D9 -  1341  1241  1415
 hCPY5%  Abs 1005662 #F585E -  1414  1467  1477
 hCPY5&  Abs 1006173 #F5A5D -  1685  1676
 hCPY5+  Abs 1006262 #F5AB6 -  1724  1713
 hCPY5,  Abs 1006047 #F59DF -  1626  1622
 hCPY5-  Abs 1005784 #F58D8 -  1485  1387  1413  1429
 hCPY5.  Abs 1004782 #F54EE -   894   891
 hCPY50  Abs 1005473 #F57A1 -  1305   894
 hCPY51  Abs 1005540 #F57E4 -  1347  1331
 hCPY52  Abs 1005561 #F57F9 -  1361  1349
 hCPY53  Abs 1005633 #F5841 -  1401  1391
 hCPY54  Abs 1005670 #F5866 -  1418  1395
 hCPY55  Abs 1005696 #F5880 -  1432  1393
 hCPY56  Abs 1005718 #F5896 -  1445  1384
 hCPY58  Abs 1006079 #F59FF -  1641  1639
 hCPY59  Abs 1006188 #F5A6C -  1694  1674  1679  1686  1688
 hCPY5?  Abs 1005482 #F57AA -  1314   930
 hCPY5a  Abs 1005536 #F57E0 -  1344  1287  1333
 hCPY5b  Abs 1005690 #F587A -  1428  1439
 hCPY5d  Abs 1005991 #F59A7 -  1599  1578
 hCPY5e  Abs 1006609 #F5C11 -  1871  1867
 hCPY5f  Abs 1006126 #F5A2E -  1666  1647
 hCPY5i  Abs 1006299 #F5ADB -  1740  1725
 hCPY5j  Abs 1005617 #F5831 -  1390  1386
 hCPY5l  Abs 1006560 #F5BE0 -  1856  1737
 hCPY5m  Abs 1006295 #F5AD7 -  1737  1733  1735  1741
=hCPY5s  Abs 1006613 #F5C15 -  1874  1707  1714  1821
 hCPY5t  Abs 1005534 #F57DE -  1343
 hCPY5x  Abs 1005825 #F5901 -  1507  1497  1500
 hCPY5y  Abs 1005917 #F595D -  1553  1545
 hCPY5z  Abs 1006225 #F5A91 -  1712  1706
 hCPY6.  Abs 1004733 #F54BD -   861   918
 hCPYE5  Abs 1006199 #F5A77 -  1700  1682
 hCPYEL  Abs 1006306 #F5AE2 -  1742  1700  1711  1720  1722
 hCPYER  Abs 1006170 #F5A5A -  1682  1651  1656
 hCPYXM  Abs 1004733 #F54BD -   867   261   770   873   884   904   913  1152
                             1302  1640
 hCPYe+  Abs 1006414 #F5B4E -  1798  1796
 hCPYe-  Abs 1006441 #F5B69 -  1807  1827
```

```
 hCPYe.  Abs 1006453 #F5B75 -  1814  1798
 hCPYeO  Abs 1006383 #F5B2F -  1786  1784
 hCPYe1  Abs 1006474 #F5B8A -  1821  1834  1836
 hCPYe2  Abs 1006489 #F5B99 -  1826  1847
 hCPYe3  Abs 1006541 #F5BCD -  1843  1829  1846
 hCPYe4  Abs 1006524 #F5BBC -  1837  1815  1819  1825  1831  1844
 hCPYeL  Abs 1006359 #F5B17 -  1767  1338  1731
 hCPYeR  Abs 1006433 #F5B61 -  1803
 hCPYel  Abs 1006437 #F5B65 -  1806  1355  1414  1549  1586
 hCPYer  Abs 1005092 #F5624 -  1068   269   379  1028  1049  1058  1060  1344
                               1803
 hCPYex  Abs 1006418 #F5B52 -  1799  1792
 hCPYt-  Abs 1005547 #F57EB -  1355  1504
 hCPYtP  Abs 1004850 #F5532 -   949
 hCPYtp  Abs 1005551 #F57EF -  1356   949  2270
 hCPYxm  Abs 1005469 #F579D -  1302  2163
=hCREAT  Abs 1003955 #F51B3 -   252
=hFINDF  Abs 1003859 #F5153 -   115
 hFNFer  Abs 1003917 #F518D -   138   118   120
=hFPROT  Abs 1007107 #F5E03 -  2258
 hPRTC0  Abs 1004619 #F544B -   748   742
 hPRTC1  Abs 1004693 #F5495 -   773   769
=hPRTCL  Abs 1004600 #F5438 -   734
 hPRTXM  Abs 1004689 #F5491 -   770   744
 hPUR20  Abs 1006864 #F5D10 -  2039  2010  2015
 hPURER  Abs 1006775 #F5CB7 -  1983  1999  2020  2050  2055
=hPURGE  Abs 1006781 #F5CBD -  1987
 hPURSC  Abs 1006818 #F5CE2 -  2019   963  1105  2291
 hPUTDR  Abs 1006849 #F5D01 -  2033  2049  2192  2341
=hRDCBF  Abs 1004228 #F52C4 -   422
=hRDNBF  Abs 1004335 #F532F -   548
=hRENAM  Abs 1006958 #F5D6E -  2131
 hRNM25  Abs 1006990 #F5D8E -  2153  2152
 hRNM30  Abs 1007013 #F5DA5 -  2169  2154
 hRNMER  Abs 1007005 #F5D9D -  2160  2149  2155  2174  2193  2195
 hRNMXM  Abs 1007009 #F5DA1 -  2163
 hRNMfx  Abs 1007000 #F5D98 -  2158  1550  2144
 hRNMsb  Abs 1007084 #F5DEC -  2200  2170
 hRNMsd  Abs 1007081 #F5DE9 -  2199   978  2139  2183
 hSEC15  Abs 1007132 #F5E1C -  2277  2269
 hSEC20  Abs 1007161 #F5E39 -  2294  2284
 hSEC25  Abs 1007172 #F5E44 -  2307  2299
 hSEC30  Abs 1007175 #F5E47 -  2312  2290  2304
 hSEC40  Abs 1007205 #F5E65 -  2329  2331
 hSECeR  Abs 1007103 #F5DFF -  2255  2264
 hSECer  Abs 1007245 #F5E8D -  2349  2255  2342
 hSECft  Abs 1007128 #F5E18 -  2270  2327
=hVER$   Abs 1003803 #F511B -    53
 hVER$1  Abs 1003857 #F5151 -    65    59
=hWRCBF  Abs 1004307 #F5313 -   498
 i/OFND  Ext                -   650  2058
 lFIB    Abs      63 #0003F -    13  2074
 lFLENh  Abs       5 #00005 -    13  1120  1169  1262  1266  1267  1459  1627
 mSDA@5  Ext                - 1875
 mSFC@5  Ext                - 1878
```

```
 o41sod  Abs       5 #00005 -    13  1250
 oACCSb  Abs      11 #0000B -    13   455   568   643   648
 oBSsod  Abs      17 #00011 -    13  1207
 oCOPYb  Abs      10 #0000A -    13   657   658
 oCPOSb  Abs      40 #00028 -    13   569   666
 oDAsod  Abs      13 #0000D -    13  1624  1692
 oDBEGb  Abs      21 #00015 -    13   568   569   658   666   678  2068  2072
                                2077
 oDEVCb  Abs      12 #0000C -    13   631
 oFBEGb  Abs      13 #0000D -    13   677   678  2067  2068  2077
 oFBF#b  Abs       2 #00002 -    13   648   657
 oFLAGh  Abs      20 #00014 -    13  1092  1097  1601
 oFLENh  Abs      32 #00020 -    13  1097  1177  1207
 oFTYPh  Abs      16 #00010 -    13  1091  1177  1860
 oIMPLh  Abs      37 #00025 -    13  1668
 oRECLb  Abs      36 #00024 -    13  2072  2073
 oRLENb  Abs      52 #00034 -    13  2073  2074
 pEOT    Ext                 - 1835
 pTERM   Ext                 - 1833
 sCARD   Abs       2 #00002 -    13   872
 sDEST   Abs       3 #00003 -    13  1642  1900  2169  2199
 sEXTDV  Abs       0 #00000 -    13   859
 sLoop?  Ext                 - 1646  1724  1814
=sNAPRS  Abs 1004277 #F52F5 -   445
 sOVERW  Ext                 -  376  1026  1285
 sPR     Abs       1 #00001 - 2282  2288
 sPRIVT  Ext                 - 2283
 sSEC    Abs       0 #00000 - 2281  2289  2303  2311
 sUNDEF  Abs       1 #00001 -    13   917
 sUNSEC  Ext                 - 2298
```

Input Parameters

  Source file name is NZ&HND::MS

  Listing file name is NZ/HND:TI:ML::-1

  Object file name is NZ%HND:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
                        TITLE  HPIL CAT <840106.1936>
F5E91                   ABS    #F5E91        TI%HP6 address (fixed)
                *
                *       N   N  ZZZZZ   &      CCC     A    TTTTT
                *       N   N      Z  & &    C   C   A A     T
                *       NN  N     Z   & &    C       A   A    T
                *       N N N    Z     &     C       A   A    T
                *       N  NN   Z     & & &  C       AAAAA    T
                *       N   N  Z      &  &   C   C   A   A    T
                *       N   N  ZZZZZ   && &   CCC    A   A    T
                *
                ****************************************************************
                ****************************************************************
                **
                ** Name:       hCAT - HPIL poll handler for the CAT statement
                **
                ** Category:   POLL
                **
                ** Purpose:
                **      Execute the CAT function for an HPIL device
                **
                ** Entry:
                **      File name in A[W], R0[3:0] (A[W]=0 if none specified)
                **      Device specifier in D[3:0], D[S]
                **      P=0
                **
                ** Exit:
                **      P=0
                **      Carry set: error (C[3:0] is error number)
                **      Carry clear:
                **        XM=0: handled (cat is finished)
                **        XM=1: not handled (not HPIL or not Filbert)
                **
                ** Calls:      CKBITL,FINDF+,SAVED1,SETCAT,BLDCAT,DSPCAT,BF2DSP,
                **             RESTD1,START,GETDR',hCATsu,CK=ATn,UTLEND,POPBUF,
                **             RPTKY,SCRLLR,FINDA,D1=AVE,ENDTAP,hCTA+,hCTA-,
                **             CSRC10,NXTENT,hCTA=,CSRC5,LSTENT,CSLC10,CSLC5
                **
                ** Uses.......
                **  Inclusive: A,B,C,D,R0,R1,R2,R3,R4,D0,D1,P,STMTD0,ST[4:0],
                **             SCRTCH[63:0],3 RSTK save fields,FUNCD0,FUNCR1,
                **             F-R0-1
                **
                ** Stk lvls:   6 (FINDF+)(hCTA+)(hCTA-)(hCTA=)
                **
                ** Detail:
                **      R3 contains the pointers to the current drive:
                **             [A] is the # of entries remaining in directory
                **               (after the current one'), including any
                **               purged entries
                **             [9:5] is the current entry number (this is the
                **               number of entries to here in the directory,
                **               including the current entry and any purged
                **               entries)
                **             [13:10] is the physical directory pointer (3 nib
```

```
**              record pointer, 1 nib offset pointer)
**            [S] is the "valid" flag - indicates whether
**              the physical directory pointer is where the
**              drive really is pointing now (0 means valid)
**
** Algorithm:
**
**      hCAT:  IF (not HPIL) or (not Filbert) THEN
**                 RETURN carry clear, XM=1 -- Not handled
**             --
**             -- This is HPIL...continue
**             --
**             IF (filename not specified) THEN CATALL
**             --
**             -- This is a specific entry
**             --
**             Find the file (FINDF+)
**             IF error then set up error, RTNSC
**             --
**             -- File found (directory entry in SCRTCH)
**             --
**             Save device address in STMTD1
**             Reserve RAM on MTHSTK for building entry
**             --
**             BLDCAT -- Build the CAT string on the stack
**             --
**             DSPCAT -- Send the string to the display
**             ---
**             GOTO hCTA35 -- Collapse the MTHSTK, RTNCC
**      ------------------------------------------------------
**      ------------------------------------------------------
**      CATALL:Save device address in STMTD1
**             Display header line (NAME...TYPE...LEN...)
**             --
**             Restore device address
**             Get directory info and first entry from drive
**             --
**             Reserve RAM on MTHSTK for building entry
**             --
**      hCTA20:Check for ATTN key pressed (if so, exit)
**             Unaddress the device as listener
**             --
**             Build the catalog entry                (BLDCAT)
**             Display the catalog entry              (DSPCAT)
**             Goto hCTA22
**             ----------
**      hCTAct --
**             -- Continue with next key
**             --
**             Unaddress talkers/listeners            (UTLEND)
**             --
**      hCTA22 Pop key from buffer (Either entry or already used)
**             --
**             Repeat key if still down               (RPTKEY)
**             --
```

```
111             **              If key not still down, get next key     (SCRLLR)
112             **              --
113             **      hCTA35  Restore device address from STMTD1      (RESTD1)
114             **              --
115             **              Set up the loop and device again        (START )
116             **              If error, goto hCTAer (clean up)
117             **              --
118             **              Set R2=R3 (R2 is temporary position)
119             **              Check keycode                           (FINDA )
120             **                Down  :goto hCTAdn
121             **                Up    :goto hCTAup
122             **                Bottom:goto hCTAbt
123             **                Top   :goto hCTAtp
124             **               Else continue
125             **              --
126             **              If keycode is not zero (CAT all) then
127             **                inhibit display scrolling
128             **              --
129             **      hCTA38  Release RAM from MTHSTK
130             **              --
131             **      hCTA39  Rewind the drive, unaddress all         (ENDTAP)
132             **              Return with carry clear, XM=0
133             **              ----------
134             **      hCTAdn  -- Down arrow
135             **              --
136             **              Get next non-purged directory entry     (hCTA+ )
137             **              --
138             **      hCTAxx  If not End_of_Directory, goto hCTAbl --Build disp
139             **              else goto hCTAct  --Ignore the down arrow
140             **              ----------
141             **      hCTAup  -- Up arrow
142             **              --
143             **              Get previous non-purged directory entry (hCTA-)
144             **              Goto hCTAxx
145             **              ----------
146             **      hCTAbt  -- gDown arrow (bottom)
147             **              --
148             **              Get next non-purged directory entry     (hCTA+ )
149             **              If not End_of_Directory, goto hCTAbt --Get next
150             **              --
151             **              -- Reached End_of_Directory...
152             **              -- ...Check if new record...if so, say not exact
153             **              --
154             **              Get the current entry                   (hCTA= )
155             **              Goto hCTA20 --Build it, display it
156             **              ----------
157             **      hCTAtp  -- gUp arrow (top)
158             **              --
159             **              If already at top, then goto hCTA&& --Redisplay it
160             **              Position to first non-purged directory entry
161             **              Goto hCTA&&  --Redisplay it
162             **
163             ** History:
164             **
165             **    Date      Programmer              Modification
```

```
166                ** --------  ----------  --------------------------------
167                ** 01/03/84      NZ       Changed RAM usage (added two RSTKBF
168                **                        levels in hCTA+c to fix bug)
169                ** 10/25/83      NZ       Updated documentation
170                ** 05/16/83      NZ       Changed CKHPIL to CKBITL, removed
171                **                        check for mass storage (done in
172                **                        CKBITL)
173                ** 04/14/83      NZ       Added call to CHKMAS
174                ** 01/14/83      NZ       Packed code (CKHPIL,FINDF+),fixed
175                **                        bug (CAT :<device>, no files on
176                **                        medium)
177                ** 12/02/82      NZ       Wrote statement & documentation
178                **
179                ****************************************************************
180                ****************************************************************
181 F5E91 7000 =hCAT   GOSUB  =CKBITL       Is this an HPIL CAT on Filbert?
182 F5E95 500          RTNNC                No...return, XM set, carry clear
183                *
184                * This IS HPIL...is it for whole device or just one file?
185                *
186 F5E98 978          ?A=0   W             Filename specified?
187 F5E9B 62           GOYES  hCATAL        No...CAT ALL
188                *
189                * This is CAT for a specific file
190                *
191 F5E9D 7000         GOSUB  =Findf+       Set up and find the file
192 F5EA1 4D7          GOC    hCATer        Not found/error
193                *
194                * Now the directory entry is in SCRTCH
195                *
196 F5EA4 DB           C=D    A
197 F5EA6 135          D1=C
198 F5EA9 8E00         GOSUBL =SAVED1       Save device address in STMTD1
           00
199 F5EAF 7E73         GOSUB  SETCAT        Reserve the stack space for entry
200 F5EB3 7ED4         GOSUB  BLDCAT        Build the CAT entry
201 F5EB7 7B47         GOSUB  DSPCAT        Display the cat entry
202 F5EBB D0           A=0    A             Clear A[B] ("keycode")
203 F5EBD 69E0         GOTO   hCTA35        Exit after cleanup
204                *-
205                *-
206 F5EC1      hCATAL
207                *
208                * This is a CAT ALL! (Device address in D[3:0])
209                *
210 F5EC1 7E50         GOSUB  hCTA10        (GOSUB to get address on RSTK)
211                *-
212                *
213                * Header string here
214                *
215 F5EC5 B1C3         NIBHEX B1C3          Cursor off - want non-readable
216 F5EC9 0202         NIBASC \   NAME \      chars
           02E4
           14D4
           5402
```

```
217 F5ED9 0202          NIBASC \   S TYP\
          0235
          0245
          9505
218 F5EE9 5402          NIBASC \E   LEN \
          0202
          C454
          E402
219 F5EF9 0202          NIBASC \   DATE \
          0244
          1445
          5402
220 F5F09 0202          NIBASC \   TIME \
          0245
          94D4
          5402
221 F5F19 DOA0          NIBHEX DOAOFF
          FF
222               *-
223               *-
224 F5F1F 6000 hCATer   GOTO   =Error       Return, set carry,err # in C[3:0]
225               *-
226               *-
227 F5F23 DB   hCTA10   C=D    A
228 F5F25 135           D1=C
229 F5F28 8E00          GOSUBL =SAVED1      Save address in STMTD1
          00
230 F5F2E 07            C=RSTK
231 F5F30 135           D1=C                Position D1 @ string
232 F5F33 8F00          GOSBVL =BF2DSP      Send the header,build the display
          000
233 F5F3A 8E00          GOSUBL =RESTD1      (Don't care about D1 any more)
          00
234 F5F40 137           CD1EX
235 F5F43 D7            D=C    A            Restore address
236 F5F45 8E00          GOSUBL =START       Set up the loop, check modes
          00
237 F5F4B 43D  hCATeR   GOC    hCATer       Error...set it up
238 F5F4E 8E00          GOSUBL =GETDR'      Get directory start, first entry
          00
239 F5F54 7162          GOSUB  hCATsu       Set up for directory
240 F5F58 42F           GOC    hCATeR       Error
241 F5F5B 8AE           ?C#0   A            Any entries?
242 F5F5E 60            GOYES  hCTA20       Yes...do them
243 F5F60 6680          GOTO   hCTAex       No...exit
244               *-
245               *-
246               *
247               * Now R3[A] is # ENTRIES remaining, R3[9:5] is current entry,
248               * R3[13:10] is current entry address
249               *
250 F5F64 8E00 hCTA20   GOSUBL =CK=ATn       Check if ATNFLG is set...
          00
251 F5F6A 531           GONC   hCTA21       ...yes it is...exit
252 F5F6D 8E00          GOSUBL =UTLEND      Unaddress the device...
```

```
          00
253 F5F73 7E14          GOSUB  BLDCAT        ...Build the catalog entry...
254 F5F77 7B86          GOSUB  DSPCAT        ...display the entry
255 F5F7B 4E0           GOC    hCTA22        Go always
256               *-
257               *-
258 F5F7E 5A7  hCTA21   GONC   hCTA38        Go always (jump out of range)
259               *-
260               *-
261 F5F81 8E00 hCTAct   GOSUBL =UTLEND       Unaddress talkers/listeners
          00
262 F5F87 443           GOC    hCTAeR        Error
263               *
264               * Pop the key, if any, out of the buffer
265               *
266 F5F8A 8F00 hCTA22   GOSBVL =POPBUF
          000
267 F5F91 8F00 hCTA25   GOSBVL =RPTKY        Repeat the last key if still down
          000
268 F5F98 490           GOC    hCTA30        (Key repeated if carry)
269 F5F9B 8F00          GOSBVL =SCRLLR       Scroll left/right
          000
270 F5FA2 968  hCTA30   ?A=0   B             Valid key?
271 F5FA5 CE            GOYES  hCTA25        No...continue
272 F5FA7 8E00 hCTA35   GOSUBL =RESTD1       Yes...process key
          00
273 F5FAD 137           CD1EX
274 F5FB0 D7            D=C    A             Restore device addr from STMTR1
275 F5FB2 D8            B=A    A             Save keycode in B[B]
276 F5FB4 8E00          GOSUBL =START        Set up the loop again
          00
277 F5FBA D4            A=B    A             Restore keycode from B[B]
278 F5FBC 495  hCTAeR   GOC    hCTAer        Error
279 F5FBF 11B           C=R3
280 F5FC2 10A           R2=C                 Use R2 as temporary position reg
281               *
282               * A[B] is the keycode of the key...check if valid CAT key
283               *
284 F5FC5 8F00          GOSBVL =FINDA
          000
285 F5FCC 00            CON(2) =k#DOWN       Down
286 F5FCE F50           REL(3) hCTAdn
287 F5FD1 00            CON(2) =k#UP         Up
288 F5FD3 A60           REL(3) hCTAup
289 F5FD6 00            CON(2) =k#BOT        Bottom
290 F5FD8 D60           REL(3) hCTAbt
291 F5FDB 00            CON(2) =k#TOP        Top
292 F5FDD B90           REL(3) hCTAtp
293 F5FE0 00            CON(2) 0             End of table
294               *
295               * This is not a valid CAT key...exit
296               *
297 F5FE2 968           ?A=0   B             Is this a single entry CAT?
298 F5FE5 41            GOYES  hCTA38        Yes...don't touch NEEDSC
299 F5FE7       hCTAex
```

```
300 F5FE7 D0          A=0    A          Clear NEEDSC (CAT :<device>)
301 F5FE9 1F00        D1=(5) =NEEDSC
          000
302 F5FF0 1590        DAT1=A 1          Clear NEEDSC to inhibit scrolling
303 F5FF4 8AA         ?C=0   A          Exit for no files on medium?
304 F5FF7 41          GOYES  hCTA39     Yes.Don't release RAM-never reserved
305 F5FF9     hCTA38
306 F5FF9 8E00        GOSUBL =D1=AVE    Set D1 to AVMEME
          00
307 F5FFF 143         A=DAT1 A          Read (AVMEME)
308 F6002 79A0        GOSUB  LC40*2     Load C[A] with 40*2 (40 bytes)
309 F6006 CA          A=A+C  A
310 F6008 141         DAT1=A A          Write out updated AVMEME
311 F600B     hCTA39
312 F600B 7000        GOSUB  =Endtap    Clean up the loop
313 F600F 20          P=     0          Ignore error from ENDTAP
314 F6011 821         XM=0
315 F6014 03          RTNCC             Return, carry clear, XM=0
316           *-
317           *-
318 F6016 80C1 hCTAer C=P    1          Save P in C[1]
319 F601A 06          RSTK=C
320 F601C 8F00        GOSBVL =POPBUF    Pop the key out of the buffer
          000
321 F6023 07          C=RSTK
322 F6025 80D1        P=C    1          Restore P from C[1]
323 F6029 65FE        GOTO   hCATer     Error exit
324           *-
325           *-
326 F602D     hCTAdn
327           *
328           * Down arrow
329           *
330 F602D 7722        GOSUB  hCTA+      Get next entry
331 F6031 44E  hCTAxx GOC    hCTAer     Error
332 F6034 8AE         ?C#0   A
333 F6037 D3          GOYES  hCTAbl     Not at end of directory...build it
334 F6039 674F        GOTO   hCTAct     End of directory...ignore key
335           *-
336           *-
337 F603D     hCTAup
338           *
339           * Up arrow
340           *
341 F603D 7772        GOSUB  hCTA-      Get previous directory entry
342 F6041 6FEF        GOTO   hCTAxx     Finish it up (error if carry)
343           *-
344           *-
345 F6045     hCTAbt
346           *
347           * (g) Down arrow [bottom]
348           *
349 F6045 7F02        GOSUB  hCTA+      Get next entry
350 F6049 4CC         GOC    hCTAer     Error...exit
351 F604C 8AE         ?C#0   A          End of directory yet?
```

```
352 F604F 6F          GOYES  hCTAbt       No...keep looking for end
353           *
354           * Check if crossed a record boundary - if so, need to re-seek
355           *
356 F6051 11B         C=R3
357 F6054 94E         ?C#0   S            Already marked as "not current"?
358 F6057 61          GOYES  hCTA&&       Yes...skip unnecessary test
359 F6059 7D56        GOSUB  Csrc10
360 F605D 7566        GOSUB  Nxtent       Check if this crossed a boundary
361 F6061 5B0         GONC   hCTA&&       No...OK as is
362 F6064 11B         C=R3                Yes...need to set C[S]="F"
363 F6067 A4E hCTA&+  C=C-1  S            (Set "not current")
364 F606A 10B         R3=C
365           *
366           * Get and build the entry now
367           *
368 F606D 73B2 hCTA&&  GOSUB  hCTA=        Get this entry
369 F6071 44A         GOC    hCTAer       Error
370 F6074 6FEE hCTAbl GOTO   hCTA20       Build it if no error
371           *-
372           *-
373 F6078     hCTAtp
374           *
375           * (g) Up arrow [top]
376           *
377 F6078 11A         C=R2                Read back pointers
378 F607B 7B26        GOSUB  Csrc5        Get entry # in C[A]
379 F607F DA          A=C    A            Save count in A[A]
380 F6081 CC          A=A-1  A            Adjust to zero-based count
381 F6083 CC          A=A-1  A            Check if this is first entry
382 F6085 47E         GOC    hCTA&&       Yes...already AT the top
383 F6088 7E16        GOSUB  Csrc5        Get pointer into C[3:0]
384 F608C 7C36 hCTAt1 GOSUB  Lstent       Back up an entry
385 F6090 7000        GOSUB  =Cslc10
386 F6094 E6          C=C+1  A            Increment "remaining" pointer
387 F6096 7026        GOSUB  Csrc10
388 F609A CC          A=A-1  A            Check if at start yet...
389 F609C 5FE         GONC   hCTAt1       ...not at start...loop back
390 F609F 7116        GOSUB  Cslc5        Set back to normal form...
391 F60A3 7906        GOSUB  C=1LC5       Set position to first record
392 F60A7 AC2         C=0    S
393 F60AA 5CB         GONC   hCTA&+       Go always...set NOT correct-->R3
394           *-
395           *-
396 F60AD D7  LC80**  D=C    A
397 F60AF 20  LC40*2  P=     0            Load C[A] with 80 (40*2)
398 F60B1 D2          C=0    A
399 F60B3 3105        LC(2)  40*2
400 F60B7 03          RTNCC               Carry clear on exit
401           ****************************************************************
402           ****************************************************************
403           **
404           ** Name:       hCAT$ - HPIL CAT$ function POLL handler
405           **
406           ** Category:   POLL
```

```
                        **
                        ** Purpose:
                        **      Execute the CAT$ function for HPIL mass storage devices
                        **
                        ** Entry:
                        **      F-R0-0 is the (saved) PC
                        **      AVMEME is the pointer to the start of string header
                        **        (The device string)
                        **      The numeric expression is on the stack after the device
                        **        string
                        **
                        ** Exit:
                        **      F-R0-0 is unchanged
                        **      Carry clear:
                        **        XM=0:
                        **          AVMEME points to the CAT$ string on the stack
                        **        XM=1:
                        **          Not HPIL/not Acc ID=16 device
                        **      Carry set:
                        **        Error (C[3:0] is error number)
                        **
                        ** Calls:      D1@AVE,POP1S,DEVPR$,CHKMAS,POP1N,D1=AVE,FLTDH,
                        **             GETDR!,hCATsu,hCTA+,BLDCAT,D1@AVS,ENDTAP,<REV$>
                        **
                        ** Uses.......
                        **  Inclusive: A-D,R0,R1,R2,R3,SCRTCH[63:0],ST[4:0],P,F-R0-1,
                        **             FUNCD0,FUNCR1
                        **
                        ** Stk lvls:   5 (GETDR!)
                        **
                        ** History:
                        **
                        **    Date      Programmer          Modification
                        **  --------    ----------    --------------------------------
                        **  01/04/84       NZ         Packed code in the vicinity of
                        **                            GOSUBL =fLTDH call, hCAT$5, and
                        **                            GOSUB =Endtap, changed RAM usage
                        **  04/14/83       NZ         Added check for D=0 after DEVPR$
                        **  12/13/82       NZ         Added routine and documentation
                        **
                        ****************************************************************
                        ****************************************************************
F60B9 21        hCAT$x  P=     1              Return, set XM: not HPIL.
F60BB 0D                P=P-1                 Clear carry, P=0
F60BD 00                RTNSXM                Set XM
                        *-
                        *-
F60BF           =hCAT$
                        *
                        * Is this an HPIL CAT$?
                        *
F60BF 7DF5              GOSUB  D1@ave         Set D1 @ start of string
F60C3 8F00              GOSBVL =POP1S         Now A[A] is string len, D1@string
      000
                        *
```

```
                        * DEVPR$ leaves D0 at the mailbox if good device spec
                        *
F60CA 8E00                      GOSUBL =DEVPR$          Get the device info
      00
F60D0 501                       GONC   hCAT$2           This is a GOOD device spec (D[A])
                        *
                        * Need to check if this is valid device spec...
                        *
F60D3 890                       ?P=    =eDSPEC          Is this a device spec error?
F60D6 3E                        GOYES  hCAT$x           Yes...return, clear carry, XM=0
F60D8 890                       ?P=    =eRANGE          Is it out of range (device spec)?
F60DB ED                        GOYES  hCAT$x           Yes...return, clear carry, XM=0
F60DD 6000              hCAT$e  GOTO   =Error           No...error
                        *-
                        *-
F60E1                   hCAT$2
                        *
                        * If D[A] is zero, then device not found
                        *
F60E1 8AB                       ?D=0   A
F60E4 5D                        GOYES  hCAT$x           Not found...return, not handled
                        *
                        * Now D[A] is the device address, D0 @ mailbox
                        *
F60E6 8E00                      GOSUBL =CHKMAS          Check if this is mass storage
      00
F60EC 4CC                       GOC    hCAT$x           Not mass storage...don't handle
                        *
                        * Now know this is a mass storage device...find the start of
                        * directory, set up for search
                        *
                        * D1 is now at the numeric value pointer -16
                        *
F60EF 17F                       D1=D1+ 16               Point to the numeric value
F60F2 8E00                      GOSUBL =POP1N           Get the value
      00
                        *
                        * Now D1 is where the string should go -16
                        *
F60F8 17F                       D1=D1+ 16
F60FB 8E00                      GOSUBL =aVE=D1          Write D1 value to AVMEME
      00
                        *
                        * A[W] is the numeric value
                        *
F6101 8E00                      GOSUBL =fLTDH           Convert to HEX
      00
                        *
                        * If XM=1, then out of range, else negative (both are null
                        * string)
                        *
F6107 533                       GONC   hCAt$5           Either negative or out of range
                        *
                        * Now A[A] is the value
                        *
```

```
511 F610A CC          A=A-1  A          Convert to base zero
512 F610C 436         GOC    hCAT$5     (Zero=null string)
513 F610F 101         R1=A              Save value in R1[A]
514           *
515           * The following call cannot be in hCATsu because of RSTK lvls
516           *
517 F6112 8E00        GOSUBL =GETDR!    Get the first entry
          00
518 F6118 7D90        GOSUB  hCATsu     Set up the drive (Position to 1st)
519 F611C 40C         GOC    hCAT$e     Error
520 F611F 8AA         ?C=0   A          No entries?
521 F6122 E4          GOYES  hCAT$5     No...exit, null string
522 F6124 111 hCAT$3  A=R1              Recall count from R1
523 F6127 CC          A=A-1  A          Check if done
524 F6129 452         GOC    hCAT$4     Yes...build the string
525 F612C 101         R1=A              Save count into R1 again
526 F612F 7521        GOSUB  hCTA+      Get next entry
527 F6133 49A         GOC    hCAT$e     Error...exit
528 F6136 8AE         ?C#0   A          End of directory?
529 F6139 BE          GOYES  hCAT$3     No...continue
530           *
531           * End of directory
532           *
533 F613B 543 hCAt$5  GONC   hCAT$5     Send null string
534           *-
535           *-
536 F613E 20  hCAT$m  P=     =eNORAM    Mem error
537 F6140 4C9         GOC    hCAT$e     Go always...error
538           *-
539           *-
540 F6143 0           CON(1) =FIXSPC    12 nibbles available here
541 F6144             BSS    12-1
542           *-
543           *-
544           *
545           * Got a good entry...save device address, build entry
546           *
547 F614F     hCAT$4
548 F614F 1F00        D1=(5) =F-R0-1    Address to save device address
          000
549 F6156 DB          C=D    A
550 F6158 145         DAT1=C A
551           *
552 F615B 7632        GOSUB  BLDCAT     Build the entry in memory
553           *
554           * Set D0 back to mailbox
555           *
556 F615F 1F00        D1=(5) =F-R0-1    Address of device address
          000
557 F6166 147         C=DAT1 A          (LC80** does a D=C A)
558 F6169 704F        GOSUB  LC80**     String is 40 bytes (80 nibbles)
559 F616D 560         GONC   hCAT$6     Go always
560           *-
561           *-
562 F6170 D2  hCAT$5  C=0    A          Length=0 (Null string)
```

```
563 F6172 20          P=     0            Must set P=0 for A=A-1 P below
564             *
565             * Now C[A] is the length of the string, AVMEME is start
566             *
567 F6174 AF0  hCAT$6  A=0    W
568 F6177 DA          A=C    A            Now A[A] is length in nibs
569 F6179 7345        GOSUB  D1@ave       Set D1 @ (AVMEME)
570             *
571             * Now A[A] is length in nibbles, D1 @ start
572             *
573 F617D BF0         ASL    W
574 F6180 BF0         ASL    W
575 F6183 A0C         A=A-1  P            Set A[0]="F"
576 F6186 1CF         D1=D1- 16           Point to string header field
577             *
578             * D1 @ intended header destination
579             *
580 F6189 137         CD1EX              Pointer in C[A]
581 F618C 06          RSTK=C
582 F618E 8E00        GOSUBL =D1@AVS     Read (AVMEMS) into D1
        00
583             *
584             * RSTK @ intended header, D1 @ (AVMEMS)
585             *
586 F6194 07          C=RSTK             (AVMEME) into C[A]
587             *
588             * D1 @ (AVMEMS), C @ intended header
589             *
590 F6196 133         AD1EX
591             *
592             * A[A] @ (AVMEMS), C[A] @ intended header
593             *
594 F6199 8B6         ?A>C   A           Room?
595 F619C 2A          GOYES  hCAT$M      No...mem error
596 F619E 133         AD1EX              Yes...OK to write it
597             *
598             * A[W] is intended header, C[A] @ intended header
599             *
600 F61A1 135         D1=C               Set D1 to start of header
601             *
602             * There is room to put this here
603             *
604 F61A4 1517        DAT1=A W           Write the string header
605             *
606             * Now set AVMEME (pointed to by D1) to the new header
607             *
608 F61A8 8E00        GOSUBL =aVE=D1     Write out new AVMEME
        00
609             *
610             * (Leave D1 @ AVMEME for REV$)
611             *
612             * Clean up the mass storage device now
613             *
614 F61AE 795E        GOSUB  hCTA39      Unaddress Talker&listener,P=0,XM=0
615 F61B2 8D00 =rEV$  GOVLNG =REV$       Reverse the string
```

```
          000
    616           ****************************************************************
    617           ****************************************************************
    618           **
    619           ** Name:       hCATsu - Subroutine for hCAT routines
    620           **
    621           ** Category:   LOCAL
    622           **
    623           ** Purpose:
    624           **      Set up for executing hCTA-, hCTA+ and BLDCAT routines
    625           **
    626           ** Entry:
    627           **      Carry clear:
    628           **        D[A] is drive address
    629           **        AVMEME points to the top of the stack
    630           **        D0 points to the HPIL mailbox
    631           **      Carry set:
    632           **        Error (will just RTNC)
    633           **
    634           ** Exit:
    635           **      Carry clear:
    636           **        C[A]=0:
    637           **          No directory entries on medium
    638           **        C[A]#0:
    639           **          R3 contains the directory pointers (see hCAT)
    640           **          AVMEME reflects the new top of stack (after reserving
    641           **            RAM for CAT)
    642           **
    643           ** Calls:       CSRC5,CSRC10,CSLC5,CSLC10,TSAV2C,R<RST2,GDIRS+,
    644           **              hCTA+C,RST2<R,TRES2C,GETMBX,SETCAT,D1=AVS,D1=AVE
    645           **
    646           ** Uses.......
    647           **  Inclusive: A[W],B[W],C[W],R2,R3,D1,P,(3 RSTK save locations)
    648           **
    649           ** Stk lvls:   3 (hCTA+c) {3 levels saved by R<RST2}
    650           **
    651           ** History:
    652           **
    653           **    Date     Programmer             Modification
    654           **  --------   ----------    ---------------------------------
    655           **  01/04/84      NZ         Reworked code around hCTA+C call
    656           **                           to reduce the number of stack
    657           **                           levels used (added R<RST2,RST2<R)
    658           **  12/14/82      NZ         Added routine and documentation
    659           **
    660           ****************************************************************
    661           ****************************************************************
    662 F61B9 400  hCATsu  RTNC                Error! (Return at once)
    663           *
    664           * Now  B[3:0] is pointer to first directory entry, D[8:5] is
    665           * number of directory records, SCRTCH is first entry,
    666           * D1 is at (=SCRTCH)+16
    667           *
    668           * Save # of directory ENTRIES remaining in R3[A], current
    669           * ENTRY number in R3[9:5]
```

```
670             *
671 F61BC AFB           C=D     W
672 F61BF 77E4          GOSUB   Csrc5           Now C[3:0] is # of records
673 F61C3 F2            CSL     A               (# records times 8 is # ENTRIES)
674 F61C5 81E           CSRB                    Now C[A] is # of ENTRIES
675 F61C8 CE            C=C-1   A               (We have the first one already)
676 F61CA 7CE4          GOSUB   Csrc10
677 F61CE D9            C=B     A
678 F61D0 7000          GOSUB   =Cslc10         C[13:10] is current dir location
679 F61D4 10A           R2=C                    # of entries, current entry-->R2
680 F61D7 10B           R3=C                    # of entries, current entry-->R3
681             *
682             * Check if the first entry is PURGED or EOD...if so, find the
683             * first non-purged entry
684             *
685 F61DA 8F00           GOSBVL =R<RST2          Save 3 RSTK levels in RAM
            000
686 F61E1 8E00          GOSUBL =GETMBX          Get the mailbox address back to D0
            00
687 F61E7 7371          GOSUB   GDIRS+          Read file type, set P=3
688 F61EB 72A0          GOSUB   hCTA+C          Check if PURGED, etc.
689 F61EF 80CE          C=P     14              Save P value in C[14]
690 F61F3 AC2           C=0     S
691 F61F6 550           GONC    hCATs1          If carry is clear, leave C[S]=0
692 F61F9 B46           C=C+1   S
693 F61FC 8E00 hCATs1   GOSUBL =TSAV2C          Save C[W] in FUNCR1 for now
            00
694 F6202 8F00          GOSBVL =RST2<R          Restore the RSTK levels
            000
695 F6209 8E00          GOSUBL =GETMBX          Restore the mailbox addr to D0
            00
696 F620F 8E00          GOSUBL =TRES2C          Restore C[W]
            00
697 F6215 80DE          P=C     14              Restore P
698 F6219 94E           ?C#0    S               Was carry set?
699 F621C 00            RTNYES                  Yes...error
700 F621E 8AA           ?C=0    A               Any valid entries?
701 F6221 F2            GOYES   hCATsx          No...exit
702 F6223 11B           C=R3
703 F6226 7084          GOSUB   Csrc5
704 F622A 7284          GOSUB   C=1LC5          Set C[A]=1, CSLC5
705 F622E 10B           R3=C                    This is the FIRST entry
706             *
707 F6231       SETCAT
708 F6231 7A7E          GOSUB   LC40*2          40 bytes = 80 nibbles
709 F6235 D5            B=C     A
710 F6237 8E00          GOSUBL =D1=AVS          Check if room for 40 bytes
            00
711 F623D 143           A=DAT1  A
712 F6240 174           D1=D1+  5               AVMEME is 5 nibbles after AVMEMS
713 F6243 147           C=DAT1  A
714 F6246 E9            C=C-B   A               Now C[A] is proposed new AVMEME
715 F6248 8B6           ?A>C    A
716 F624B 70            GOYES   SETenm          No memory
717             *
```

```
                * There IS room for this
                *
F624D 145               DAT1=C A            Write out the (temp) AVMEME
F6250 03        hCATsx  RTNCC               Return, carry clear
                *-
                *-
F6252 8C00      SETenm  GOLONG =NORAMe      No memory
      00
                ****************************************************************
                ****************************************************************
                **
                ** Name:       hCTA+ - Go forward 1 non-purged entry
                **
                ** Category:   LOCAL
                **
                ** Purpose:
                **      Move one non-purged directory entry forward from
                **      current position
                **
                ** Entry:
                **      D0 points to the mailbox, D[X] is device address
                **      R2 is current position pointers, R3 is old pointers
                **
                ** Exit:
                **      Carry clear:
                **        C[A]=0: No more directory entries
                **        C[A]#0: R3 updated to current pointers
                **      Carry set:
                **        Error (P=error code)
                **
                ** Calls:      CSRC10,NXTENT,SEEKRD,CSRC5,GDIRSB
                **
                ** Uses.......
                **  Exclusive:      C[W],R2,R3
                **  Inclusive: A[A],C[W],R2,R3,D1,P
                **
                ** Stk lvls:   5 (GDIRSB)
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------    ---------------------------------
                **  01/04/84      NZ          Packed to install bug fix for CAT
                **                            on a medium with the first file
                **                            purged
                **  12/10/82      NZ          Added documentation
                **
                ****************************************************************
                ****************************************************************
F6258 11A       hCTA+   C=R2
                *
                * Down arrow key (C[W] is R2 contents)
                *
F625B 8AA               ?C=0    A
F625E E4                GOYES   hCTA+x      Exit...already at end of directory
```

```
772             *
773             * Have NOT reached EOD yet
774             *
775 F6260 94A           ?C=0   S              Is the medium at that record?
776 F6263 41            GOYES  hCTA+2         Yes...don't need to SEEK
777             *
778             * Need to position to that record
779             *
780 F6265 7154          GOSUB  Csrc10         C[3:1] is record #, [0] is BP
781 F6269 7954          GOSUB  Nxtent         Set to NEXT record
782 F626D 7CF0          GOSUB  SEEKRD         Seek to the record & read it
783 F6271 400           RTNC                  Error
784 F6274 11A  hCTA+1   C=R2
785             *
786             * Now the medium is positioned at the record specified
787             *
788 F6277 8AA  hCTA+2   ?C=0   A              End of directory?
789 F627A 22            GOYES  hCTA++         Yes...exit, mark end of directory
790 F627C CE            C=C-1  A              No...decrement the count
791 F627E 7824          GOSUB  Csrc5
792 F6282 7224          GOSUB  C+1RC5         Increment current location
793 F6286 7C34          GOSUB  Nxtent         Go to next entry
794             *
795             * GDIRSB sets R2 to C after CSLC10, C[S]=0
796             *
797 F628A 78B0          GOSUB  GDIRSB         Get directory entry, set up
798 F628E 400           RTNC                  Error
799             *
800             * Now the entry is in SCRTCH, C[3:0] is type (byte-reversed)
801             *
802 F6291 91A  hCTA+C   ?C=0   WP             Purged entry?
803 F6294 0E            GOYES  hCTA+1         Yes...get next one
804 F6296 B16           C=C+1  WP
805 F6299 541           GONC   hCTA+!         Done: P=3, carry clear
806             *
807             * End of directory...set count=0, position flag=false
808             *   (Set position=last good position from R3)
809             *
810 F629C 11B  hCTA++   C=R3                  End of directory
811 F629F D2            C=0    A
812 F62A1 AC2  hCTA&t   C=0    S
813 F62A4 A4E           C=C-1  S              Set R3[S]#0 (not at current record)
814 F62A7 10B           R3=C                  Set # of entries remaining=0
815 F62AA D2            C=0    A              (Needed for hCTA&t entry)
816 F62AC 03    hCTA+x  RTNCC                 (Carry clear, C[A]=0)
817             *-
818             *-
819 F62AE 11A  hCTA+!   C=R2
820 F62B1 10B           R3=C                  Update the pointers
821 F62B4 E6            C=C+1  A              Insure that C[A]#0 for exit cond
822 F62B6 03            RTNCC
823             ****************************************************************
824             ****************************************************************
825             **
826             ** Name:       hCTA- - Move back one directory entry
```

```
827             ** Name:       hCTA= - Get the current directory entry
828             **
829             ** Category:   LOCAL
830             **
831             ** Purpose:
832             **      hCTA-:Move back one non-purged directory entry
833             **      hCTA=:Read in the current directory entry
834             **
835             ** Entry:
836             **      D0 points to the mailbox, D[X] is device address
837             **      R2 is current directory pointers, R3 is old pointers
838             **
839             ** Exit:
840             **      Carry clear:
841             **        C[A]=0: Beginning of directory reached
842             **        C[A]#0: SCRTCH[63:0] is the new entry
843             **                R3 is updated to current directory entry
844             **      Carry set:
845             **        Error (P=error code)
846             **
847             ** Calls:      CSRC5,CSLC5,NXTENT,LSTENT,SEEKRD,GDIRSB
848             **
849             ** Uses.......
850             **  Exclusive: A[A],C[W],R2,R3,D1,P
851             **  Inclusive: A[A],C[W],R2,R3,D1,P
852             **
853             ** Stk lvls:   5 (GDIRSB)
854             **
855             ** History:
856             **
857             **    Date      Programmer            Modification
858             **  --------    ----------    --------------------------------
859             **  01/04/84       NZ         Packed to install bug fix (see CAT)
860             **  01/03/84       NZ         Moved the RTNC after SEEKRD to be
861             **                            before the C=B A (Was destroying
862             **                            the error number in C[0])
863             **  01/24/83       NZ         Changed R2[A] to include purged
864             **                            entries
865             **  12/10/82       NZ         Added documentation
866             **
867             ********************************************************************
868             ********************************************************************
869 F62B8 11A  hCTA-   C=R2
870 F62BB 79E3        GOSUB  C+1RC5        Increment # of entries left
871 F62BF CE          C=C-1  A             Decrement to previous entry
872 F62C1 8AA         ?C=0   A             At top already?
873 F62C4 A4          GOYES  hCTA-3        Yes...set R3 to first entry
874 F62C6 70E3        GOSUB  Csrc5
875 F62CA 10A         R2=C                 Save counts in R2 for now
876 F62CD 25          P=     15-10         Point to C[S], CSRC5'ed twice
877 F62CF DA          A=C    A             Save entry in A[A]
878 F62D1 90E         ?C#0   P             Is this the current position?
879 F62D4 21          GOYES  hCTA-1        No...need to SEEK that record
880 F62D6 7CE3        GOSUB  Nxtent        Check if this was the last entry
881 F62DA 4B0         GOC    hCTA-1        Was last...need to SEEK
```

```
882 F62DD D6          C=A    A          Check if was FIRST entry
883 F62DF 79E3        GOSUB  Lstent     Go back 1 entry (record)
884 F62E3 5F0         GONC   hCTA-2     Still in same record
885             *
886 F62E6 D6   hCTA-1 C=A    A          Get the entry location back again
887 F62E8 70E3        GOSUB  Lstent     Go back 1 entry (for position)
888             *
889             * Now C[3:1] is the correct record #
890             *
891             * Go to that record
892             *
893 F62EC 7D70        GOSUB  SEEKRD     Seek to that record, read it
894 F62F0 400         RTNC              Error
895 F62F3      hCTA-2
896             *
897             * Now medium is positioned to the correct record
898             *
899 F62F3 11A         C=R2
900 F62F6 72D3        GOSUB  Lstent     Set C[3:0] to the last entry
901 F62FA 7840        GOSUB  GDIRSB     Get directory entry, set P=3
902 F62FE 400         RTNC              Error
903             *
904             * D1 @ (=SCRTCH)+20, P=3, C[3:0] is type (byte-reversed)
905             *
906 F6301 91A         ?C=0   WP         Purged?
907 F6304 4B          GOYES  hCTA-      Yes...try next entry
908             *
909             * Good entry (Cannot get EOD with up-arrow)
910             *
911 F6306 11A         C=R2
912 F6309 10B         R3=C              Set R3 to the current pointer
913 F630C 03          RTNCC
914             *-
915             *-
916 F630E 11B  hCTA-3 C=R3
917 F6311 7593        GOSUB  Csrc5
918 F6315 CE          C=C-1  A
919 F6317 8AA         ?C=0   A
920 F631A 29          GOYES  hCTA+x     Started at beginning..leave as is
921 F631C 7093        GOSUB  C=1LC5     Indicate at FIRST entry in CAT
922 F6320 608F        GOTO   hCTA&t     Set R3[S]#0, continue
923             *-
924             *-
925 F6324 11B  hCTA=  C=R3
926 F6327 7F83        GOSUB  Csrc10     Get current entry into C[3:0]
927 F632B 25          P=     15-10      Point to R3[S], shifted 10
928 F632D 90A         ?C=0   P          Is it correct?
929 F6330 61          GOYES  GDIRSB     Yes...just read that entry
930 F6332 D5          B=C    A          (Save entry info in B[3:0])
931 F6334 7530        GOSUB  SEEKRD     No...SEEK to the record, read it
932             *
933             * Before restoring entry information to C[3:0], check for error
934             *
935 F6338 400         RTNC              Error if carry set
936 F633B D9          C=B    A          (Restore entry info to C[3:0])
```

```
                        CR3EX               Save C[3:0] in R3, fetch R3-->C
                        C=0    S            Current record is positioned
                        CR3EX               Restore C[3:0], R3
                *
                * Fall through to GDIRSB
                *
                ********************************************************************
                ********************************************************************
                **
                ** Name:       GDIRSB - Subroutine to get a directory entry
                **
                ** Category:   LOCAL
                **
                ** Purpose:
                **      Save location, get directory entry, check file type
                **
                ** Entry:
                **      C[3:0] is the directory pointer
                **      D0 points to the mailbox
                **      D[X] is the device address
                **
                ** Exit:
                **      Carry clear:
                **        P=3, C[3:0]=file type (C[B] is high byte of type)
                **      Carry set:
                **        Error (P=error code)
                **
                ** Calls:      CSLC10,GETDR+
                **
                ** Uses.......
                **  Exclusive: A[A],C[W],R2,D1,P
                **  Inclusive: A[A],C[W],R2,D1,P
                **
                ** Stk lvls:   4 (GETDR+)
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **  --------   ----------   --------------------------------
                **  01/04/84      NZ        Added GDIRS+ entry point
                **  12/09/82      NZ        Added routine & documentation
                **
                ********************************************************************
                ********************************************************************
                *
                * Code above falls into this routine
                *
F6346 DA    =GDIRSB A=C     A               Copy entry to A[A]
F6348 7000          GOSUB   =Cslc10         Restore R2 to correct orientation
F634C AC2           C=0     S               (At correct record)
F634F 10A           R2=C                    Set R2 again
                *
                * Now A[3:0] is the CORRECT pointer for this file
                *
F6352 814           ASRC
```

```
 992 F6355 8E00          GOSUBL =GETDR+        Set byte pointer, read entry
           00
 993 F635B 400           RTNC                  Error
 994 F635E 1F00 GDIRS+   D1=(5) (=SCRTCH)+20   Position to TYPE bytes
           000
 995 F6365 15F3          C=DAT1 4
 996 F6369 23            P=     3
 997 F636B 03            RTNCC                 Leave C[3:0]=type, P=3
 998             ********************************************************************
 999             ********************************************************************
1000             **
1001             ** Name:       SEEKRD - Seek to a record, then read it
1002             **
1003             ** Category:   PILI/O
1004             **
1005             ** Purpose:
1006             **      Seek a record on the mass memory device and read it
1007             **
1008             ** Entry:
1009             **      C[3:1] is the record # desired
1010             **      DO points to the mailbox
1011             **      D[X] is the device address
1012             **
1013             ** Exit:
1014             **      Carry clear:
1015             **        P=0, record has been read into buffer 0 of device
1016             **      Carry set: Error (P=error #)
1017             **        Error (P,C[0] are the error code)
1018             **
1019             ** Calls:      TSTAT,SEEKA,DDT,TSTATA
1020             **
1021             ** Uses.......
1022             **  Exclusive: A[A],C[W],P
1023             **  Inclusive: A[A],C[W],P
1024             **
1025             ** Stk lvls:   3 (TSTAT)(SEEKA)(TSTATA)
1026             **
1027             ** History:
1028             **
1029             **    Date     Programmer             Modification
1030             **  --------   ----------    ---------------------------------
1031             **  12/09/82      NZ         Added routine & documentation
1032             **
1033             ********************************************************************
1034             ********************************************************************
1035 F636D       =SEEKRD
1036             *
1037             * Go to the record, but check status first
1038             *
1039 F636D D0           A=0    A
1040 F636F F6           CSR    A
1041 F6371 ABA          A=C    X              A[A] is now record #
1042 F6374 8E00         GOSUBL =TSTAT         Check device status first
           00
1043 F637A 400          RTNC                  Error
```

```
1044 F637D 7000         GOSUB  =Seeka        Go to that record
1045 F6381 400          RTNC
1046 F6384 20           P=     =Read
1047 F6386 8E00         GOSUBL =DDT          Read the data from the device
           00
1048 F638C 400          RTNC
1049 F638F 8C00         GOLONG =TSTATA       (Device is already talker)
           00
1050            ****************************************************************
1051            ****************************************************************
1052            **
1053            ** Name:       BLDCAT - Build CAT text, given directory entry
1054            **
1055            ** Category:   LOCAL
1056            **
1057            ** Purpose:
1058            **      Build the CAT[$] string on the [MATH] stack, using the
1059            **      directory entry in SCRTCH[63:0]
1060            **
1061            ** Entry:
1062            **      SCRTCH contains the directory entry for the file
1063            **
1064            ** Exit:
1065            **      Carry clear, CAT text on stack, AVMEME at CAT text
1066            **
1067            ** Calls:      D1@AVE,TSAVDO,BLANKC,SWAPO1,GT2BYT,FTYPF#,HTODX,
1068            **             WRTASC,GETBYT,GT2BYO,A-MULT,TRESDO
1069            **
1070            ** Uses.......
1071            **  Exclusive: A[W],B[W],C[W],D[S],R0,D1,P
1072            **  Inclusive: A[W],B[W],C[W],D[S],R0,D1,P,FUNCDO
1073            **
1074            ** Stk lvls:   3 (FTYPF#)
1075            **
1076            ** History:
1077            **
1078            **    Date     Programmer            Modification
1079            **  --------   ----------   --------------------------------
1080            **  12/06/82      NZ        Wrote routine and documentation
1081            **
1082            ****************************************************************
1083            ****************************************************************
1084 F6395 7723 =BLDCAT GOSUB  D1@ave        Set D1 to start of string
1085            *
1086            * Now D1 is at start of CAT build area, SCRTCH contains the
1087            * directory entry for the desired CAT
1088            *
1089            * Save D0 in FUNCD0 (restore on exit)
1090            *
1091 F6399 8E00         GOSUBL =TSAVDO
           00
1092 F639F 1B00         D0=(5) =SCRTCH
           000
1093 F63A6 1567         C=DAT0 W             Read in first 8 chars of name
1094 F63AA 16F          D0=D0+ 16            Skip first 8 input chars
```

```
1095 F63AD 1557          DAT1=C W            Write out the first 8 chars
1096 F63B1 17F           D1=D1+ 16
1097 F63B4 146           C=DAT0 A            Read last 2 chars
1098 F63B7 163           D0=D0+ 4            Skip last 2 input chars
1099 F63BA 15D3          DAT1=C 4            Write last 2 chars
1100 F63BE 173           D1=D1+ 4
1101               *
1102               * Now the name is written...blank, security, blank next
1103               *
1104 F63C1 8E00          GOSUBL =BLANKC      Get blanks in C[W]
          00
1105               *
1106               * Blank out the rest of the text now
1107               *
1108 F63C7 133           AD1EX
1109 F63CA 131           D1=A                Save D1 in A[A]
1110 F63CD 2B            P=     16-5
1111 F63CF 15DB BLDC10   DAT1=C 6*2          Clear the remaining 30 bytes
1112 F63D3 17B           D1=D1+ 6*2            in chunks of 6 bytes
1113 F63D6 0C            P=P+1
1114 F63D8 56F           GONC   BLDC10
1115 F63DB 131           D1=A                Restore D1
1116 F63DE 175           D1=D1+ 6            Skip to file type field
1117               *
1118               * D1 points to the file type byte in header
1119               *
1120               * D0 is still at the file type in SCRTCH
1121               *
1122 F63E1 AF2           C=0    W            Must clear high nibs for HTODX
1123 F63E4 7312          GOSUB  SWAP01       Swap D0, D1
1124 F63E8 8E00          GOSUBL =GT2BYT      Read in 2 bytes (type) at D1
          00
1125 F63EE 7902          GOSUB  SWAP01       Swap D0, D1
1126               *
1127               * D0 is now at start of start address field, D1 is still at
1128               * text "type" field
1129               *
1130 F63F2 AFA           A=C    W            File type into A[A]
1131 F63F5 7000          GOSUB  =fTYPF#      Read the file type
1132               *
1133               * If carry set, found the type; C[A], B[A] @ entry, B[S] = #
1134               *
1135 F63F9 4A2           GOC    BLDC30       Found a file type table with this
1136               *
1137               * This is an unknown type...leave security blank, print
1138               * type in ASCII digits (Type is in A[W])
1139               *
1140 F63FC AC3           D=0    S            Use D[S] as the SIGN of file type
1141 F63FF D6            C=A    A            Check if A[3:0] is #8000 or more
1142 F6401 F2            CSL    A
1143 F6403 C6            C=C+C  A            If carry, then this is negative
1144 F6405 5A0           GONC   BLDC20       Non-negative...continue
1145               *
1146               * This is negative...change sign field to 1
1147               *
```

```
1148 F6408 B47          D=D+1  S
1149 F640B 23           P=     3
1150 F640D B98          A=-A   WP            Negative of file type
1151 F6410 8E00 BLDC20  GOSUBL =HTODX        Convert to decimal
           00
1152 F6416 24           P=     4             B[W]<=32768 to get here
1153 F6418 7732         GOSUB  WRTASC        Write digits, suppress leading 0's
1154 F641C D1           B=0    A             Set B[A]=0...type not known
1155 F641E 171          D1=D1+ 2             Skip a blank between type, length
1156 F6421 5B3          GONC   BLDC40        Go always...continue with length
1157            *-
1158            *-
1159 F6424      BLDC30
1160            *
1161            * B[A] is pointer to file type, B[S] is the protection
1162            * D1 at file type text area
1163            *
1164 F6424 A4D          B=B-1  S             Always at LEAST 1 from FTYPF#
1165            *
1166            * Now B[S] is the protection, base zero
1167            *
1168 F6427 1C3          D1=D1- 4             Point to the protection byte
1169 F642A AC9          C=B    S             Read protection type
1170 F642D A46          C=C+C  S             Double it for bytes
1171 F6430 BCA          C=-C   S             Negate it for offset from C[S]
1172 F6433 80DF         P=C    15            Set P=offset from C[S]
1173 F6437 3702         LCASC  \EPS \        C[B] gets proper value
           3505
           54
1174 F6441 14D          DAT1=C B             Write out the security code
1175 F6444 173          D1=D1+ 4             Back to file type text area
1176            *
1177            * Now ready to output the file type
1178            *
1179 F6447 D9           C=B    A
1180 F6449 137          CD1EX                D1-->type entry
1181 F644C 174          D1=D1+ 5             Skip to ASCII for file type
1182 F644F 15B9         A=DAT1 10            Read the type...
1183 F6453 137          CD1EX                ...restore true D1...
1184 F6456 1599         DAT1=A 10            ...and write the type
1185 F645A 17B          D1=D1+ 12            (Skip to length field)
1186 F645D      BLDC40
1187            *
1188            * Now continue at the length field
1189            *
1190 F645D 8AD          ?B#0   A             Is the type known?
1191 F6460 F1           GOYES  BLDC50        Yes...continue
1192            *
1193            * Type is unknown...use size in records
1194            *
1195 F6462 167          D0=D0+ 8             Skip the start of file field
1196            *
1197            * D0 is at the length of file in records
1198            *
1199 F6465 7291 BLDC45  GOSUB  SWAP01        Swap D0, D1 (D1 @ start of field)
```

```
1200 F6469 24          P=     4
1201 F646B AF2         C=0    W
1202 F646E 8E00        GOSUBL =GETBYT       Read 5 bytes into C[9:0]
          00
1203 F6474 AE2         C=0    B             Throw away low byte
1204               *
1205               * C[W] is now the file size in bytes (records * 256)
1206               *
1207 F6477 7081        GOSUB  SWAP01        Restore D1 from D0
1208 F647B 6590        GOTO   BLDC60        File size (bytes) in C[W]
1209               *-
1210               *-
1211 F647F D9    BLDC50 C=B    A
1212 F6481 E6          C=C+1  A             Skip create code
1213 F6483 134         D0=C                 D0 points to start of entry
1214 F6486 AF2         C=0    W
1215 F6489 1564        C=DAT0 S             Read copy code from type table
1216 F648D 161         D0=D0+ 2             Point to offset to data
1217 F6490 14E         C=DAT0 B             Read offset to data value
1218 F6493 AF5         B=C    W             Copy to B[W]
1219 F6496 1B00        D0=(5) (=SCRTCH)+56  Point to implementation bytes
          000
1220 F649D 94E         ?C#0   S             Copy code zero?
1221 F64A0 42          GOYES  BLDC52        No...check further
1222               *
1223               * Copy code zero...length is (IMPL)-(oDATA)+(lFLEN)
1224               *
1225 F64A2 15A5        A=DAT0 6             Read in the length field
1226 F64A6 20          P=     0
1227 F64A8 3100        LC(2)  =lFLENh       Length of FLEN field
1228 F64AC 25          P=     5
1229 F64AE A12         C=C+A  WP
1230 F64B1 B19         C=C-B  WP            Subtract offset to data
1231 F64B4 550         GONC   BLDC51
1232 F64B7 AF2         C=0    W             If less than zero, set =0
1233 F64BA        BLDC51
1234               *
1235               * Now C[W] is the length in nibbles
1236               *
1237 F64BA B76         C=C+1  W             Add one to round UP if odd
1238 F64BD 81E         CSRB                 Convert to bytes
1239 F64C0 6050        GOTO   BLDC60        Done (size in C[W])
1240               *-
1241               *-
1242 F64C4        BLDC52
1243               *
1244               * Check further on the copy code
1245               *
1246 F64C4 A46         C=C+C  S             Copy code 8?
1247 F64C7 550         GONC   BLDC54        Not copy code 8...continue
1248               *
1249               * Copy code 8...use length in records to display size
1250               *
1251 F64CA 480         GOC    BLDC5?        Go always
1252               *-
```

```
1253                    *-
1254 F64CD A46  BLDC54  C=C+C  S            Copy code 4 (LIF1)?
1255 F64D0 5C0          GONC   BLDC56       No...keep checking
1256                    *
1257                    * This is LIF1...use length in records
1258                    *
1259 F64D3 1B00 BLDC57  DO=(5) (=SCRTCH)+32  Length in records
             000
1260 F64DA 4A8          GOC    BLDC45       Go always (use record length)
1261                    *-
1262                    *-
1263 F64DD A46  BLDC56  C=C+C  S            Copy code 2 (41C data file)?
1264 F64E0 591          GONC   BLDC58       No...must be TITAN data file
1265                    *
1266                    * 41C (SDATA) data file
1267                    *
1268 F64E3 7411         GOSUB  SWAP01
1269 F64E7 8E00         GOSUBL =GT2BYO      Read 2 bytes (size in registers)
             00
1270 F64ED 7A01         GOSUB  SWAP01
1271 F64F1 BF2          CSL    W
1272 F64F4 81E          CSRB                Multiply by 8 bytes/register
1273 F64F7 591          GONC   BLDC60       Go always (Size in C[W])
1274                    *-
1275                    *-
1276 F64FA      BLDC58
1277                    *
1278                    * TITAN data file
1279                    *
1280 F64FA 15E3         C=DAT0 4            Read # of records
1281 F64FE 163          DO=DO+ 4            Position to record length
1282 F6501 AF0          A=0    W            Clear high nibs of A[W]
1283 F6504 15A3         A=DAT0 4            Read record length
1284 F6508 8E00         GOSUBL =A-MULT      Leaves result in A[W]
             00
1285                    *
1286                    * A[W] is now the length
1287                    *
1288 F650E AF6          C=A    W            Copy to C[W]
1289 F6511 AFA  BLDC60  A=C    W            Copy size to A[W]
1290                    *
1291                    * Convert size to decimal...
1292                    *
1293 F6514 8E00         GOSUBL =HTODX       Result in B[W]
             00
1294 F651A 2F           P=     15
1295 F651C 90D  BLDC65  ?B#0   P
1296 F651F 90           GOYES  BLDC70       Non-zero digit
1297 F6521 0D           P=P-1
1298 F6523 58F          GONC   BLDC65       Go unless B[W]=0
1299 F6526 20           P=     0            Indicate 1 digit
1300 F6528      BLDC70
1301                    *
1302                    * Now B[WP] is the decimal value of size
1303                    *
```

```
1304 F6528 80CF          C=P    15
1305 F652C AC5           B=C    S              Save (WP) in B[S]
1306 F652F AC3           D=0    S              Set D[S]=0 (for WRTASC)
1307 F6532 20            P=     0
1308 F6534 3500          LC(6)  \MK\~0         C[B] is current mode
          B4D4
1309 F653C 2F            P=     15
1310 F653E 305           LC(1)  5
1311 F6541 985  BLDC71   ?B<C   P              Are there more than 5 digits?
1312 F6544 62            GOYES  BLDC75         No...continue
1313             *
1314             * More than 5 digits...
1315             *  ...if 5-8 digits, represent as xxxxK
1316             *  ...if >8 digits, represent as xxxxM
1317             *
1318 F6546 BF6           CSR    W
1319 F6549 F6            CSR    A              Shift next #0/K/M into C[B]
1320 F654B BF5           BSR    W
1321 F654E BF5           BSR    W
1322 F6551 05            SETDEC
1323 F6553 A05           B=B+B  P              Rounding digit
1324 F6556 BF5           BSR    W
1325 F6559 550           GONC   BLDC72
1326 F655C B75           B=B+1  W              Add one for rounding
1327 F655F 04   BLDC72   SETHEX
1328             *
1329             * For the case of >8 digits, this will execute this code a
1330             * third time.  The ?B<C  P test will fail, as B[12] will be
1331             * zero from BSR   W's that have been done the first 2 times
1332             *
1333 F6561 2C            P=     15-3           Point to current length location
1334 F6563 308           LC(1)  8              Are there more than 8 digits?
1335 F6566 6ADF          GOTO   BLDC71         Check for more than 8 digits
1336             *-
1337             *-
1338 F656A       BLDC75
1339             *
1340             * Now C[B] is the tail character, B[A] is the value, P#0
1341             *
1342 F656A DA            A=C    A              Copy C[B] to A[B]
1343 F656C 24            P=     4              5 digits unless C[B]#0, then 4
1344 F656E 96A           ?C=0   B              Is the suffix (Null)?
1345 F6571 40            GOYES  BLDC77         Yes...5 digits
1346 F6573 0D            P=P-1                 No...4 digits
1347 F6575 7ADO BLDC77   GOSUB  WRTASC         Write the ASCII to the text area
1348 F6579 968           ?A=0   B              Is suffix character zero?
1349 F657C 80            GOYES  BLDC78         Yes...go on
1350 F657E 149           DAT1=A B              No...write the suffix character
1351 F6581 171           D1=D1+ 2              Skip suffix character
1352 F6584 171  BLDC78   D1=D1+ 2              Point to date/time field
1353             *
1354             * Now D1 @ start of date field of text
1355             *
1356 F6587 1B00          D0=(5) (=SCRTCH)+40   Point to time/date field
          000
```

```
1357                *
1358                * Next seven lines are to convert YYMMDD to MMDDYY
1359                *
1360 F658E 15E5          C=DAT0 6            Read in YYMMDD
1361 F6592 163           D0=D0+ 4            Point to DD
1362 F6595 14C           DAT0=C B            Write out YY
1363 F6598 183           D0=D0- 4
1364 F659B BF6           CSR    W
1365 F659E F6            CSR    A
1366 F65A0 15C3          DAT0=C 4            Write out MM DD
1367                *
1368 F65A4 20            P=     0
1369 F65A6 AF2           C=0    W
1370 F65A9 3103          LCASC  \0\          Set high nib of A[B] for digits
1371 F65AD DA            A=C    A
1372 F65AF 39F2          LCASC  \ : //\      Separator for MM/DD/YY HH:MM
             F202
             A302
1373 F65BB 160  BLDC80   D0=D0+ 1
1374 F65BE 15A0          A=DAT0 1            Read first digit
1375 F65C2 149           DAT1=A B            Write first digit
1376 F65C5 171           D1=D1+ 2
1377 F65C8 180           D0=D0- 1            Point to second digit...
1378 F65CB 15A0          A=DAT0 1            ...read it...
1379 F65CF 161           D0=D0+ 2            (skip to next digit)
1380 F65D2 149           DAT1=A B            ...and write second digit
1381 F65D5 171           D1=D1+ 2
1382 F65D8 14D           DAT1=C B            Write the separator
1383 F65DB 171           D1=D1+ 2
1384 F65DE BF6           CSR    W
1385 F65E1 BF6           CSR    W            Shift in next separator
1386 F65E4 96E           ?C#0   B            Done yet?
1387 F65E7 4D            GOYES  BLDC80       No...continue
1388                *
1389                * Set D1 back to start of text...
1390                *
1391 F65E9 2B            P=     16-5         Loop 5 times
1392 F65EB 1CF  BLDC90   D1=D1- 16           (16*5 nibbles in text)
1393 F65EE 0C            P=P+1
1394 F65F0 5AF           GONC   BLDC90
1395 F65F3 8E00          GOSUBL =TRESD0      Restore D0 from FUNCD0
             00
1396 F65F9 03            RTNCC               Return with carry clear
1397                *-
1398                *-
1399 F65FB 136  =SWAP01  CD0EX               Swap D0, D1
1400 F65FE 137           CD1EX
1401 F6601 136           CD0EX
1402 F6604 01            RTN                 Don't change carry
1403                ****************************************************************
1404                ****************************************************************
1405                **
1406                ** Name:       DSPCAT - Display a CAT text string from @ D1
1407                **
1408                ** Category:   LOCAL
```

```
1409                **
1410                ** Purpose:
1411                **      Send 40 bytes (starting at D1) to the display
1412                **
1413                ** Entry:
1414                **      D1 @ start of data
1415                **
1416                ** Exit:
1417                **      P=0
1418                **
1419                ** Calls:      D0=FR0,SWAP01,CKINF-,SEND20,CURSFL,CRLFND
1420                **
1421                ** Uses.......
1422                **  Inclusive: A-D,R0,D0,D1,all FUNCxx except FUNCR0,STMTR0,P
1423                **
1424                ** Stk lvls:   5 (CURSFL)
1425                **
1426                ** History:
1427                **
1428                **    Date      Programmer              Modification
1429                **  --------    ----------    --------------------------------
1430                **  12/06/82      NZ          Added code and documentation
1431                **
1432                ****************************************************************
1433                ****************************************************************
1434 F6606 20    =DSPCAT P=     0
1435 F6608 8E00         GOSUBL =D0=FR0       Set D0=FUNCR0
            00
1436 F660E 1527         A=DAT0 W
1437 F6612 100          R0=A                 Save FUNCR0 in R0
1438 F6615 72EF         GOSUB  SWAP01        Save D1 in D0
1439 F6619 8F00         GOSBVL =CKINF-       Set up display, check info
            000
1440 F6620 77DF         GOSUB  SWAP01        Restore D1
1441 F6624 110          A=R0                 Restore FUNCR0 from R0 to A[W]...
1442 F6627 8E00         GOSUBL =D0=FR0       ...set D0 @ FUNCR0...
            00
1443 F662D 1507         DAT0=A W             ...and write to FUNCR0
1444 F6631 133          AD1EX                Get D1 into A[A]
1445 F6634 D2           C=0    A
1446 F6636 3182         LC(2)  40            Send 40 bytes
1447 F663A DE           ACEX   A             A[A]=length in bytes, C[A]=start
1448 F663C D7           D=C    A             D[A]=start of string
1449                *
1450                * D[A] is at start of string, A[A] is length
1451                *
1452 F663E 8F00         GOSBVL =SEND20       Send it, ignore width
            000
1453                *
1454                * Set no delay, cursor far left
1455                *
1456 F6645 8F00         GOSBVL =CURSFL       Cursor far left
            000
1457 F664C 8D00         GOVLNG =CRLFND       Cr, Lf, no delay (builds display)
            000
```

```
1458            ****************************************************************
1459            ****************************************************************
1460            **
1461            ** Name:       WRTASC - Write out a decimal number in ASCII
1462            **
1463            ** Category:   GETUTL
1464            **
1465            ** Purpose:
1466            **      Write a decimal number from B[WP] to RAM @ D1
1467            **
1468            ** Entry:
1469            **      D1 at intended destination field (initialized to \ \)
1470            **      P is the first digit location in B to be considered
1471            **      B[WP] is the value
1472            **      D[S] is sign of value (D[S]=0:positive; else negative)
1473            **
1474            ** Exit:
1475            **      D1 past the last digit
1476            **      P=1 (NOTE THIS!)
1477            **      Carry clear
1478            **
1479            ** Calls:      None
1480            **
1481            ** Uses.......
1482            **  Inclusive: C[S,WP],D1,P
1483            **
1484            ** Stk lvls:   0
1485            **
1486            ** Detail:
1487            **      Write out the digits, starting with the first non-zero
1488            **      digit (if B[W]=0, write a single zero out)
1489            **
1490            ** History:
1491            **
1492            **    Date     Programmer             Modification
1493            **  --------   ----------   --------------------------------
1494            **  12/06/82      NZ        Added documentation
1495            **
1496            ****************************************************************
1497            ****************************************************************
1498 F6653 90D  =WRTASC ?B#0   P            Is leading digit non-zero?
1499 F6656 F0          GOYES  WRTA10        Yes...found a non-zero digit
1500 F6658 171         D1=D1+ 2             No...skip to next text location
1501 F665B 0D          P=P-1                Decrement P (if zero, will carry)
1502 F665D 55F         GONC   WRTASC        Go unless B[WP] was zero
1503 F6660 20          P=     0             B[WP] was zero...output 1 digit
1504 F6662 1C1         D1=D1- 2             (Back up the last add)
1505            *
1506 F6665 94B  WRTA10  ?D=0   S            Check the sign field
1507 F6668 51          GOYES  WRTA20        Positive...NO sign output
1508 F666A 137         CD1EX                Negative...output a leading "-"
1509 F666D 1DD2        D1=(2) \-\           Put a "-" in C[B], leave P as is
1510 F6671 137         CD1EX
1511 F6674 1C1         D1=D1- 2
1512 F6677 14D         DAT1=C B             Write the leading sign
```

```
1513 F667A 171          D1=D1+ 2            Point back to first digit
1514 F667D      WRTA20
1515            *
1516            * Now P is the first digit, D1 at text location for first digit
1517            *
1518 F667D 80CF         C=P    15           Save the pointer in C[S]
1519            *
1520 F6681 80DF WRTA30  P=C    15           Get pointer to P again
1521 F6685 A89         C=B    P            Copy B[P] to C[P]
1522 F6688 890  WRTA40  ?P=    0
1523 F668B A0           GOYES  WRTA50       Digit is in C[0] now
1524 F668D B96          CSR    WP
1525 F6690 0D           P=P-1
1526 F6692 55F          GONC   WRTA40       Go always
1527            *-
1528            *-
1529 F6695 21   WRTA50  P=     1
1530 F6697 303          LCHEX  3            High nibble for ASCII #
1531 F669A 14D          DAT1=C B            Write the digit
1532 F669D 171          D1=D1+ 2
1533 F66A0 A4E          C=C-1  S            Check if more digits
1534 F66A3 5DD          GONC   WRTA30       Not done yet...continue
1535            *
1536            * Have finished writing B[W] out in ASCII
1537            *
1538 F66A6 03           RTNCC
1539            *-
1540            *-
1541 F66A8 E6   C+1RC5  C=C+1  A            Add 1 to C[A], CSRC5
1542 F66AA 8C00 Csrc5   GOLONG =CSRC5
         00
1543            *-
1544            *-
1545 F66B0 D2   C=1LC5  C=0    A            Set C[A]=1, CSLC5
1546 F66B2 E6          C=C+1  A
1547 F66B4 8C00 Cslc5   GOLONG =CSLC5
         00
1548            *-
1549            *-
1550 F66BA 8C00 Csrc10  GOLONG =CSRC10
         00
1551            *-
1552            *-
1553 F66C0 8C00 D1@ave  GOLONG =D1@AVE
         00
1554            *-
1555            *-
1556 F66C6 8C00 Nxtent  GOLONG =NXTENT
         00
1557            *-
1558            *-
1559 F66CC 8C00 Lstent  GOLONG =LSTENT
         00
1560 F66D2              END
```

```
 A-MULT  Ext                  -  1284
 BF2DSP  Ext                  -   232
 BLANKC  Ext                  -  1104
 BLDC10  Abs 1008591 #F63CF   -  1111  1114
 BLDC20  Abs 1008656 #F6410   -  1151  1144
 BLDC30  Abs 1008676 #F6424   -  1159  1135
 BLDC40  Abs 1008733 #F645D   -  1186  1156
 BLDC45  Abs 1008741 #F6465   -  1199  1260
 BLDC50  Abs 1008767 #F647F   -  1211  1191
 BLDC51  Abs 1008826 #F64BA   -  1233  1231
 BLDC52  Abs 1008836 #F64C4   -  1242  1221
 BLDC54  Abs 1008845 #F64CD   -  1254  1247
 BLDC56  Abs 1008861 #F64DD   -  1263  1255
 BLDC58  Abs 1008890 #F64FA   -  1276  1264
 BLDC5?  Abs 1008851 #F64D3   -  1259  1251
 BLDC60  Abs 1008913 #F6511   -  1289  1208  1239  1273
 BLDC65  Abs 1008924 #F651C   -  1295  1298
 BLDC70  Abs 1008936 #F6528   -  1300  1296
 BLDC71  Abs 1008961 #F6541   -  1311  1335
 BLDC72  Abs 1008991 #F655F   -  1327  1325
 BLDC75  Abs 1009002 #F656A   -  1338  1312
 BLDC77  Abs 1009013 #F6575   -  1347  1345
 BLDC78  Abs 1009028 #F6584   -  1352  1349
 BLDC80  Abs 1009083 #F65BB   -  1373  1387
 BLDC90  Abs 1009131 #F65EB   -  1392  1394
=BLDCAT  Abs 1008533 #F6395   -  1084   200   253   552
 C+1RC5  Abs 1009320 #F66A8   -  1541   792   870
 C=1LC5  Abs 1009328 #F66B0   -  1545   391   704   921
 CHKMAS  Ext                  -   484
 CK=ATn  Ext                  -   250
 CKBITL  Ext                  -   181
 CKINF-  Ext                  -  1439
 CRLFND  Ext                  -  1457
 CSLC5   Ext                  -  1547
 CSRC10  Ext                  -  1550
 CSRC5   Ext                  -  1542
 CURSFL  Ext                  -  1456
 Cslc10  Ext                  -   385   678   985
 Cslc5   Abs 1009332 #F66B4   -  1547   390
 Csrc10  Abs 1009338 #F66BA   -  1550   359   387   676   780   926
 Csrc5   Abs 1009322 #F66AA   -  1542   378   383   672   703   791   874   917
 D0=FR0  Ext                  -  1435  1442
 D1=AVE  Ext                  -   306
 D1=AVS  Ext                  -   710
 D1@AVE  Ext                  -  1553
 D1@AVS  Ext                  -   582
 D1@ave  Abs 1009344 #F66C0   -  1553   458   569  1084
 DDT     Ext                  -  1047
 DEVPR$  Ext                  -   463
=DSPCAT  Abs 1009158 #F6606   -  1434   201   254
 Endtap  Ext                  -   312
 Error   Ext                  -   224   472
 F-R0-1  Ext                  -   548   556
 FINDA   Ext                  -   284
 FIXSPC  Ext                  -   540
```

```
 Findf+  Ext                    -   191
 GDIRS+  Abs 1008478 #F635E     -   994   687
=GDIRSB  Abs 1008454 #F6346     -   984   797   901   929
 GETBYT  Ext                    -  1202
 GETDR!  Ext                    -   238   517
 GETDR+  Ext                    -   992
 GETMBX  Ext                    -   686   695
 GT2BYO  Ext                    -  1269
 GT2BYT  Ext                    -  1124
 HTODX   Ext                    -  1151  1293
 LC40*2  Abs 1007791 #F60AF     -   397   308   708
 LC80**  Abs 1007789 #F60AD     -   396   558
 LSTENT  Ext                    -  1559
 Lstent  Abs 1009356 #F66CC     -  1559   384   883   887   900
 NEEDSC  Ext                    -   301
 NORAMe  Ext                    -   724
 NXTENT  Ext                    -  1556
 Nxtent  Abs 1009350 #F66C6     -  1556   360   781   793   880
 POP1N   Ext                    -   493
 POP1S   Ext                    -   459
 POPBUF  Ext                    -   266   320
 R<RST2  Ext                    -   685
 RESTD1  Ext                    -   233   272
 REV$    Ext                    -   615
 RPTKY   Ext                    -   267
 RST2<R  Ext                    -   694
 Read    Ext                    -  1046
 SAVED1  Ext                    -   198   229
 SCRLLR  Ext                    -   269
 SCRTCH  Ext                    -   994  1092  1219  1259  1356
=SEEKRD  Abs 1008493 #F636D     -  1035   782   893   931
 SEND20  Ext                    -  1452
 SETCAT  Abs 1008177 #F6231     -   707   199
 SETenn  Abs 1008210 #F6252     -   724   716
 START   Ext                    -   236   276
=SWAP01  Abs 1009147 #F65FB     -  1399  1123  1125  1199  1207  1268  1270  1438
                                   1440
 Seeka   Ext                    -  1044
 TRES2C  Ext                    -   696
 TRESD0  Ext                    -  1395
 TSAV2C  Ext                    -   693
 TSAVD0  Ext                    -  1091
 TSTAT   Ext                    -  1042
 TSTATA  Ext                    -  1049
 UTLEND  Ext                    -   252   261
 WRTA10  Abs 1009253 #F6665     -  1506  1499
 WRTA20  Abs 1009277 #F667D     -  1514  1507
 WRTA30  Abs 1009281 #F6681     -  1520  1534
 WRTA40  Abs 1009288 #F6688     -  1522  1526
 WRTA50  Abs 1009301 #F6695     -  1529  1523
=WRTASC  Abs 1009235 #F6653     -  1498  1153  1347  1502
 aVE=D1  Ext                    -   498   608
 eDSPEC  Ext                    -   468
 eNORAM  Ext                    -   536
 eRANGE  Ext                    -   470
```

```
 fLTDH   Ext                 -   502
 fTYPF#  Ext                 -  1131
=hCAT    Abs 1007249 #F5E91 -   181
=hCAT$   Abs 1007807 #F60BF -   454
 hCAT$2  Abs 1007841 #F60E1 -   475   464
 hCAT$3  Abs 1007908 #F6124 -   522   529
 hCAT$4  Abs 1007951 #F614F -   547   524
 hCAT$5  Abs 1007984 #F6170 -   562   512   521   533
 hCAT$6  Abs 1007988 #F6174 -   567   559
 hCAT$e  Abs 1007837 #F60DD -   472   519   527   537
 hCAT$m  Abs 1007934 #F613E -   536   595
 hCAT$x  Abs 1007801 #F60B9 -   449   469   471   480   485
 hCATAL  Abs 1007297 #F5EC1 -   206   187
 hCATeR  Abs 1007435 #F5F4B -   237   240
 hCATer  Abs 1007391 #F5F1F -   224   192   237   323
 hCATs1  Abs 1008124 #F61FC -   693   691
 hCATsu  Abs 1008057 #F61B9 -   662   239   518
 hCATsx  Abs 1008208 #F6250 -   721   701
 hCAt$5  Abs 1007931 #F613B -   533   507
 hCTA&&  Abs 1007725 #F606D -   368   358   361   382
 hCTA&+  Abs 1007719 #F6067 -   363   393
 hCTA&t  Abs 1008289 #F62A1 -   812   922
 hCTA+   Abs 1008216 #F6258 -   766   330   349   526
 hCTA+!  Abs 1008302 #F62AE -   819   805
 hCTA++  Abs 1008284 #F629C -   810   789
 hCTA+1  Abs 1008244 #F6274 -   784   803
 hCTA+2  Abs 1008247 #F6277 -   788   776
 hCTA+C  Abs 1008273 #F6291 -   802   688
 hCTA+x  Abs 1008300 #F62AC -   816   771   920
 hCTA-   Abs 1008312 #F62B8 -   869   341   907
 hCTA-1  Abs 1008358 #F62E6 -   886   879   881
 hCTA-2  Abs 1008371 #F62F3 -   895   884
 hCTA-3  Abs 1008398 #F630E -   916   873
 hCTA10  Abs 1007395 #F5F23 -   227   210
 hCTA20  Abs 1007460 #F5F64 -   250   242   370
 hCTA21  Abs 1007486 #F5F7E -   258   251
 hCTA22  Abs 1007498 #F5F8A -   266   255
 hCTA25  Abs 1007505 #F5F91 -   267   271
 hCTA30  Abs 1007522 #F5FA2 -   270   268
 hCTA35  Abs 1007527 #F5FA7 -   272   203
 hCTA38  Abs 1007609 #F5FF9 -   305   258   298
 hCTA39  Abs 1007627 #F600B -   311   304   614
 hCTA=   Abs 1008420 #F6324 -   925   368
 hCTAbl  Abs 1007732 #F6074 -   370   333
 hCTAbt  Abs 1007685 #F6045 -   345   290   352
 hCTAct  Abs 1007489 #F5F81 -   261   334
 hCTAdn  Abs 1007661 #F602D -   326   286
 hCTAeR  Abs 1007548 #F5FBC -   278   262
 hCTAer  Abs 1007638 #F6016 -   318   278   331   350   369
 hCTAex  Abs 1007591 #F5FE7 -   299   243
 hCTAt1  Abs 1007756 #F608C -   384   389
 hCTAtp  Abs 1007736 #F6078 -   373   292
 hCTAup  Abs 1007677 #F603D -   337   288
 hCTAxx  Abs 1007665 #F6031 -   331   342
 k#BOT   Ext                 -   289
```

```
 k#DOWN  Ext                  -   285
 k#TOP   Ext                  -   291
 k#UP    Ext                  -   287
 lFLENh  Ext                  -  1227
=rEV$    Abs 1008050 #F61B2 -   615
```

Input Parameters

  Source file name is NZ&CAT::MS

  Listing file name is NZ/CAT:TI:ML::-1

  Object file name is NZ%CAT:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *
 2            *       N   N  ZZZZZ   &     III  00000  RRRR
 3            *       N   N      Z  & &     I   0   0  R   R
 4            *       NN  N     Z   & &     I   0   0  R   R
 5            *       N N N    Z     &      I   0   0  RRRR
 6            *       N  NN   Z     & & &   I   0   0  R R
 7            *       N   N  Z      &  &    I   0   0  R  R
 8            *       N   N  ZZZZZ   && &  III  00000  R   R
 9            *
10            *
11                    TITLE  I/O(NEW Mailbox)<831101.2117>
12 F66D2              ABS    #F66D2        TI%HP6 address (fixed)
13            *
14            * Mailbox locations and bits
15            *
16            =oOUTST EQU    6
17            =oOUTHS EQU    7
18            MAV     EQU    0
19            NRD     EQU    1
20            *
21            =oINHS  EQU    8
22            =oINST  EQU    9
23            *
24            * Local handshake bits
25            *
26            sPUTX   EQU    0
27            sGETX   EQU    0
28            sCHKER  EQU    1             This MUST not be same bit as MAV!
29            *
30            * End of equates
31            *
32            ****************************************************************
33            ****************************************************************
34            **
35            ** Name:       READIT,READSU - Read into RAM from loop
36            **
37            ** Category:   PILI/O
38            **
39            ** Purpose:
40            **      Read data, given a buffer to put it into, and a count
41            **      of how many bytes to enter
42            **
43            ** Entry:
44            **      D0 points to mailbox
45            **      D1 points to the input buffer
46            **      A[A] is the number of bytes to read
47            **      A[5] is the converstion type for Diamond
48            **
49            **      READSU: C[5:0] is start message and count
50            **      READIT: the conversation is started
51            **
52            ** Exit:
53            **      Carry clear: D1 points past the last character
54            **                   A[A] is zero
55            **      Carry set:   Error...A[A] is the number of bytes left
```

```
                **                      in the buffer
                **                    If P= =ePIL, C[6:0], [S] is status msg
                **                      from Diamond ([S] has been doubled)
                **                    Else C[W] is undefined
                **
                ** Calls:      PUTE,GETX,FRAME-
                **
                ** Uses.......
                **  Exclusive: A[5:0],C[W],D1,P
                **  Inclusive: A[5:0],C[W],D1,P,ST[3:0]
                **
                ** Stk lvls:   1 (FRAME-)(GETX)(PUTE)
                **
                ** Algorithm:
                **      READSC:Save conversation descriptor in A[5:0]
                **      READS+:Start the conversation                (PUTE)
                **      READIT:If no more data to read (A[A]=0) then RTNCC
                **             Get a message from Diamond            (GETX)
                **             If not data, check the message:       (FRAME-)
                **               If EOT or terminator match, GOTO READS+
                **                  else error
                **             (data)
                **             If P#0 then write out 3 data bytes
                **               else write out 1 byte
                **             Increment D1 past data just written
                **             GOTO READIT
                **
                ** History:
                **
                **    Date      Programmer          Modification
                **  --------    ----------    --------------------------------
                **  09/20/83      NZ          Updated documentation
                **  04/07/83      NZ          Changed to handle EOT, terminator
                **  11/23/82      NZ          Added documentation
                **
                ********************************************************************
                ********************************************************************
                *
                * START THE CONVERSATION...
                *
F66D2 25        =READSU P=      5             Save start conversation in A[5]
F66D4 A9A               A=C     WP
F66D7 7A74      READS+  GOSUB   PUTE
F66DB 400               RTNC
                *
                * ...READ THE DATA
                *
F66DE 8A8       =READIT ?A=0    A
F66E1 26                GOYES   READI9        Done!
F66E3 7E50              GOSUB   GETX
F66E7 462               GOC     READER        Error if carry
F66EA 890               ?P=     0
F66ED 94                GOYES   READI3        Single byte transfer
                * must be a triple-byte transfer
F66EF 132               ADOEX
```

```
111 F66F2 182          D0=D0- 3
112 F66F5 132          ADOEX
113 F66F8 4C0          GOC    READI2        Read too many! (Can "never" be)
114 F66FB 15D5         DAT1=C 6
115 F66FF 175          D1=D1+ 6
116 F6702 5BD          GONC   READIT        GO ALWAYS...loop back for more
117             *-
118             * If fall through, ERROR!
119             *-
120 F6705 20   READI2  P=     0             If here, A[A] is <0...too far!
121 F6707 300          LC(1)  =eUNEXP
122 F670A 20           P=     =ePIL
123 F670C 02           RTNSC
124             *-
125             *-
126 F670E 890  READER  ?P=    =eABORT       Is this an ABORT?
127 F6711 00           RTNYES               Yes...error!
128 F6713 8E00         GOSUBL =FRAME-       Decode what it is
          00
129 F6719 890          ?P=    =pSTATE       Is this "Current state"?
130 F671C 21           GOYES  BADRD1        Yes...error in C[4]
131 F671E AF6          C=A    W             Can destroy C[W] now!
132 F6721 890          ?P=    =pEOT         Was it an EOT?
133 F6724 3B           GOYES  READS+        Yes...restart it
134 F6726 890          ?P=    =pTERM        Was it a terminator char?
135 F6729 EA           GOYES  READS+        Yes...reset count, continue
136 F672B 59D          GONC   READI2        No...error!
137             *-
138             *-
139 F672E 80D4 BADRD1  P=C    4             Fetch the error nibble...
140 F6732 6221         GOTO   GETST2        Go always (CPEX 0,P= ePIL,RTNSC)
141             *-
142             *-
143 F6736 CC   =READI3 A=A-1  A             Single byte transfer
144             * can never carry...since A[A] was not zero!
145 F6738 14D          DAT1=C B
146 F673B 171          D1=D1+ 2
147 F673E 5F9          GONC   READIT        GO ALWAYS...Loop back for more
148             *-
149             *-
150             * if fall through, than ERROR! (can "never" happen)
151 F6741 02           RTNSC
152             *-
153             *-
154 F6743 03   READI9  RTNCC
155             ********************************************************************
156             ********************************************************************
157             **
158             ** Name:       GETX - Fast DATA input routine
159             **
160             ** Category:   PILI/O
161             **
162             ** Purpose:
163             **      Fast data input routine...read DATA bytes as quickly
164             **      as possible
```

```
165                **
166                ** Entry:
167                **      D0 points to the mailbox
168                **      Conversation is set up and started
169                **
170                ** Exit:
171                **      If carry clear:
172                **          P=0: C[B] is a data byte
173                **          P=2: C[5:0] is three byte quantity; C[B] is first!
174                **      If carry set:
175                **          P=0: C[6:0] is message, C[S] is status*2
176                **          P#0: Aborted (P= =eABORT)
177                **
178                ** Calls:       None
179                **
180                ** Uses.......
181                **  Inclusive: C[W],P,ST[3:0]
182                **
183                ** Stk lvls:   0
184                **
185                ** History:
186                **
187                **    Date      Programmer              Modification
188                **  --------    ----------    ---------------------------------
189                **  09/20/83       NZ         Updated documentation
190                **  04/07/83       NZ         Changed exit condition for not
191                **                            data to P=0, carry set
192                **  03/02/83       NZ         Changed to check for sERROR bit
193                **  02/15/83       NZ         Changed error for ATTN to eABORT
194                **  11/23/82       NZ         Added documentation
195                **
196                ********************************************************************
197                ********************************************************************
198 F6745 167    =GETX    D0=D0+ oINHS
199 F6748 0B     GETX1    CSTEX
200 F674A 15E0           C=DAT0 1              Read handshake
201 F674E 0B             CSTEX
202 F6750 860            ?ST#1  MAV
203 F6753 62             GOYES  GETXE          No message yet!
204                * message available!
205 F6755 160    GETX2    D0=D0+ (oINST)-(oINHS)
206 F6758 15E6           C=DAT0 7
207 F675C 816            CSRC                  Now C[S] is the status nibble
208 F675F 188            D0=D0- oINST
209 F6762 A46            C=C+C  S              Check if Three-byte transfer...
210 F6765 560            GONC   GETX3          Not triple byte
211 F6768 22             P=     2              Indicate triple byte!
212 F676A 03             RTNCC
213                *-
214                *-
215 F676C        GETX3
216                *
217                * Either single byte or not data!
218                *
219 F676C 80D2           P=C    2              Check opcode
```

```
220 F6770 888          ?P#    8            Data?
221 F6773 20           GOYES  GETX4        No...
222 F6775 20   GETX4   P=     0            YES!!!...flag it as 1 byte!
223 F6777 01           RTN                 Carry clear if OK, else set
224            *-
225            *-
226            *
227 F6779 850  GETXE   ST=1   sGETX        This is GETX
228 F677C 851  GETx.   ST=1   sCHKER       DO check error bit
229 F677F      GETXNE
230            *
231            * First check for error bit set
232            *
233 F677F 861          ?ST=0  sCHKER       Should I check error?
234 F6782 81           GOYES  GETx.N       No...check attn
235 F6784 160          DO=DO+ (oINST)-(oINHS) Point to error nib
236 F6787 15E0         C=DATO 1            Read nibble into C[0]
237 F678B 180          DO=DO- (oINST)-(oINHS) Put it back where it was
238 F678E 0B           CSTEX
239 F6790 870          ?ST=1  =sERROR      Is the error bit set?
240 F6793 20           GOYES  GETx..       (Set carry if set)
241 F6795 0B   GETx..  CSTEX
242 F6797 432          GOC    GETxE        Error bit set...error!
243            *
244            * Now check if the Attn key has been pressed
245            *
246 F679A 860  GETx.N  ?ST=0  =Attn
247 F679D 52           GOYES  GETX.        Not waiting for Attn...continue
248            *
249            * Check if "ATTN" key has been pressed TWICE
250            *
251 F679F 136          CDOEX               Save DO in C[A]
252 F67A2 1B00         DO=(5) =ATNFLG
           000
253 F67A9 1564         C=DATO S
254 F67AD 134          DO=C
255 F67B0 94A          ?C=0   S
256 F67B3 F0           GOYES  GETX.        If not ATTN, keep trying
257 F67B5 B46          C=C+1  S            Check if hit more than once...
258 F67B8 490          GOC    GETX.        No...continue
259 F67BB 187  GETxE   DO=DO- oINHS        Yes...reset DO.
260 F67BE 20           P=     =eABORT      Aborted by ATTN key or error!
261 F67C0 02           RTNSC
262            *-
263            *-
264 F67C2 861  GETX.   ?ST=0  sCHKER       Is it GETNE?
265 F67C5 E0           GOYES  GETNO        This is GETNE
266 F67C7 860          ?ST=0  sGETX        Is it GETX or GET?
267 F67CA F1           GOYES  GET1         This is GET
268 F67CC 6B7F         GOTO   GETX1        This is GETX
269            ****************************************************************
270            ****************************************************************
271            **
272            ** Name:       GET - Get a message from Diamond
273            ** Name:       GETNE - Get a message without checking error bit
```

```
                **
                ** Category:   PILI/O
                **
                ** Purpose:
                **
                ** Entry:
                **      DO points to the HPIL mailbox
                **
                ** Exit:
                **      Carry clear:
                **        Contents of mailbox in C[7:0]
                **        Handshake nibble in ST[3:0]
                **        Status nibble in C[S]
                **      Carry set:
                **        Error (P=error number)
                **
                ** Calls:      None
                **
                ** Uses.......
                **  Inclusive: C[W],ST[3:0] (P only if error)
                **
                ** Stk lvls:   0
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------    ---------------------------------
                **  09/20/83       NZ         Updated documentation
                **  03/07/83       NZ         Added GETNE
                **  03/02/83       NZ         Modified to share code with GETX
                **  11/23/82       NZ         Added documentation
                **
                ********************************************************************
                ********************************************************************
F67D0 167   =GETNE  DO=DO+ oINHS
F67D3 0B    GETNO   CSTEX
F67D5 15E0          C=DATO 1              Read handshake
F67D9 0B            CSTEX
F67DB 841           ST=0   sCHKER         Clear the sCHKER bit for GETXNE
F67DE 860           ?ST#1  MAV
F67E1 E9            GOYES  GETXNE         No message, don't check error
F67E3 521           GONC   GET2           Go always...message...get it!
                *-
                *-
F67E6 167   =GET    DO=DO+ oINHS
                *
F67E9 0B    GET1    CSTEX
F67EB 15E0          C=DATO 1              READ HANDSHAKE NIBBLE
F67EF 0B            CSTEX
F67F1 860           ?ST#1  MAV            IS MESSAGE AVAILABLE?
F67F4 11            GOYES  GET9           NO...CONTINUE WAITING
                *
                * A message is available
                *
F67F6 160   GET2    DO=DO+ (oINST)-(oINHS)
```

```
 329 F67F9 15E6         C=DAT0 7              READ THE MESSAGE
 330 F67FD 816          CSRC                  Put the status nibble in C[S]
 331 F6800 188          D0=D0- oINST
 332 F6803 03           RTNCC
 333             *-
 334             *-
 335             *
 336             * Waiting for frame available...check Attn flag
 337             *
 338 F6805 840  GET9    ST=0   sGETX          This is GET, not GETX
 339 F6808 637F         GOTO   GETx.          Check if Attn set
 340             ******************************************************************
 341             ******************************************************************
 342             **
 343             ** Name:       GETHS2 - Get the second Diamond handshake nibble
 344             **
 345             ** Category:   PILI/O
 346             **
 347             ** Purpose:
 348             **      Get the software status nibble from the HPIL mailbox
 349             **
 350             ** Entry:
 351             **      D0 points to the HPIL mailbox
 352             **
 353             ** Exit:
 354             **      Software status nibble in ST[3:0], carry clear
 355             **
 356             ** Calls:      None
 357             **
 358             ** Uses.......
 359             **  Inclusive: ST[3:0]
 360             **
 361             ** Stk lvls:   0
 362             **
 363             ** History:
 364             **
 365             **    Date     Programmer             Modification
 366             **  --------   ----------    ---------------------------------
 367             **  11/23/82      NZ         Added documentation
 368             **
 369             ******************************************************************
 370             ******************************************************************
 371 F680C 0B   =GETHS2 CSTEX                Save C[X] in ST[11:0]
 372 F680E 168          D0=D0+ oINST
 373 F6811 15E0         C=DAT0 1              Read software status in C[0]
 374 F6815 188          D0=D0- oINST
 375             *
 376             * PIL info in ST[3:0], C unchanged
 377             *
 378 F6818 0B           CSTEX
 379 F681A 01           RTN
 380             ******************************************************************
 381             ******************************************************************
 382             **
 383             ** Name:       GETST  - Get status from Diamond
```

```
                ** Name:       GETERR - Get error message from Diamond
                ** Name:       GETST- - Read status message from mailbox with-
                **                      out checking the error bit
                **
                ** Category:   PILI/O
                **
                ** Purpose:
                **      Get status/error message from Diamond
                **
                ** Entry:
                **      DO points to the HPIL mailbox
                **
                ** Exit:
                **      Carry clear: PIL status in C[X], error # in C[3]
                **                   P=0
                **      Carry set: Error (# in P,C[0])
                **
                ** Calls:      PUTC+N,GETNE,FRAME+
                **
                ** Uses.......
                **  Exclusive: C[W],        P
                **  Inclusive: C[W],ST[3:0],P
                **
                ** Stk lvls:   1 (PUTC+N)(GETNE)(FRAME+)
                **
                ** History:
                **
                **    Date     Programmer            Modification
                **   --------  ----------   ---------------------------------
                **   09/20/83     NZ        Updated documentation
                **   03/19/83     NZ        Changed both routines so that
                **                          they wait for a status message to
                **                          be sent by Diamond, instead of
                **                          erroring out with P=ePIL,C=eUNEXP
                **   03/07/83     NZ        Changed GETERR again...to use
                **                          new routines PUTC+N and GETNE
                **   03/04/83     NZ        Modified GETERR to wait for MAV
                **                            before calling GET (otherwise
                **                            GET will check the sERROR bit
                **                            while waiting and abort out!)
                **   02/03/83     NZ        Modified GETERR to return with
                **                            error if Diamond error # is #0
                **   11/23/82     NZ        Added documentation
                **
                ********************************************************************
                ********************************************************************
F681C 20        =GETST  P=      0
F681E 3100              LC(2)   =mSTATS         Request status
F6822 6900              GOTO    GETER0
                *-
                *-
F6826 20        =GETERR P=      0
F6828 3100              LC(2)   =mERSTS
F682C 7D43      GETER0  GOSUB   PUTC+N          Write it
F6830 400               RTNC
```

```
 439 F6833 799F =GETST- GOSUB  GETNE        Get the message-don't check error
 440 F6837 400          RTNC
 441 F683A 8E00         GOSUBL =FRAME+
           00
 442 F6840 880          ?P#    =pSTATE      Is it a current state?
 443 F6843 0F           GOYES  GETST-       No...get another one
 444 F6845 80D4         P=C    4            Check if error # is zero
 445 F6849 BB2          CSL    X            Move all status bits to C[3:1]
 446 F684C 880          ?P#    0            Zero?
 447 F684F 60           GOYES  GETER3       No...error!
 448 F6851 F6           CSR    A            Move all status bits into C[X]
 449 F6853 03           RTNCC               Done!
 450             *-
 451             *-
 452 F6855       GETST2
 453 F6855 80F0 GETER3  CPEX   0
 454 F6859 20           P=     =ePIL        PIL Error
 455 F685B 02           RTNSC
 456             ****************************************************************
 457             ****************************************************************
 458             **
 459             ** Name:       GETD - Get data message
 460             ** Name:       GETEND - Get EOT message
 461             **
 462             ** Category:   PILI/O
 463             **
 464             ** Purpose:
 465             **      Read a data/EOT message from Diamond
 466             **
 467             ** Entry:
 468             **      Expecting data/EOT from the mailbox
 469             **      DO points to the mailbox
 470             **
 471             ** Exit:
 472             **      Carry clear:
 473             **        Frame in C[X]
 474             **        Frame type in C[S]
 475             **      Carry set:
 476             **        GETD: Not a data frame/aborted/error bit set
 477             **        GETEND: Not an EOT frame/aborted/error bit set
 478             **
 479             ** Calls:      GET,FRAME+
 480             **
 481             ** Uses.......
 482             **  Exclusive: C
 483             **  Inclusive: C,ST[3:0]  (P only if error)
 484             **
 485             ** Stk lvls:   1 (GET)(FRAME+)
 486             **
 487             ** History:
 488             **
 489             **    Date     Programmer          Modification
 490             **  --------   ----------   --------------------------------
 491             **  09/20/83      NZ        Updated documentation
 492             **  11/23/82      NZ        Added documentation
```

```
493                **
494                **********************************************************************
495                **********************************************************************
496 F685D 758F =GETD   GOSUB  GET           Get frame
497 F6861 400          RTNC                 Error
498 F6864 8E00 =CHECKD GOSUBL =FRAME+       Check what kind of frame it is
          00
499 F686A 880          ?P#    =pDATA        DATA?
500 F686D 20           GOYES  GETD1         No...set carry
501 F686F 80FF  GETD1  CPEX   15            Yes...Carry clear!
502 F6873 500          RTNNC
503 F6876 6E8E         GOTO   READI2
504                *-
505                *-
506 F687A 786F =GETEND GOSUB  GET           Get frame
507 F687E 400          RTNC                 Error
508 F6881 8E00 =CHKEND GOSUBL =FRAME+       Decode frame
          00
509 F6887 880          ?P#    =pEOT         END?
510 F688A 5E           GOYES  GETD1         No...set carry
511 F688C 52E          GONC   GETD1         Yes...clear carry
512                **********************************************************************
513                **********************************************************************
514                **
515                ** Name:       GETID - Read 8 bytes data into A after YTMLL
516                ** Name:       READRG - Read 8 bytes data into the A register
517                ** Name:       GETID+ - Read 8 bytes data into A after YTML
518                **
519                ** Category:   PILI/O
520                **
521                ** Purpose:
522                **      Read up to 8 bytes of data from a device and put it
523                **      into A[W] (GETID and GETID+ strip Cr and trailing
524                **      characters)
525                **
526                ** Entry:
527                **      D[X] is address of the device
528                **      DO @ mailbox
529                **
530                **      READRG: Conversation is already set up
531                **
532                ** Exit:
533                **      Carry clear:
534                **        Up to 8 bytes in A[W], number of bytes in D[S]
535                **        P=0
536                **      Carry set:
537                **        Error (other than device not ready)
538                **        P,C[0]= Error #
539                **
540                ** Calls:      YTML(GETID+),YTMLL(GETID),PUTE,GETX,FRAME-
541                **
542                ** Uses.......
543                **  Exclusive: A[W],C[W],D[S],D[13],P
544                **  Inclusive: A[W],C[W],D[S],D[13],P
545                **
```

```
546            ** Stk lvls:   2 (YTMLL)(YTML)  (READRG uses only 1 level)
547            **
548            ** History:
549            **
550            **    Date      Programmer             Modification
551            **  --------    ----------    ---------------------------------
552            **  09/20/83       NZ         Updated documentation
553            **  09/01/83       NZ         Added check for P= =eABORT at GOC
554            **                            from GETX (fix of SPOLL&STANDBY
555            **                            bug)
556            **  03/09/83       NZ         Added check for not changing #
557            **                            bytes received if strip is false
558            **  03/03/83       NZ         Added check for READRG to not
559            **                            strip trailing Cr
560            **  11/23/82       NZ         Added documentation
561            **
562            ********************************************************************
563            ********************************************************************
564 F688F 8E00 =GETID+ GOSUBL =YTML          D[X] is talker, I am listener
          00
565 F6895 551          GONC   GETIDO        If no errors
566 F6898 02           RTNSC                Error!
567            *-
568            *-
569 F689A 2D   =READRG P=     13
570 F689C A83          D=0    P             Clear "strip returns" flag
571 F689F 6B10         GOTO   READRg
572            *-
573            *-
574 F68A3 20   =GETID  P=     0
575 F68A5 8E00         GOSUBL =YTMLL        D[X] is talker, I am listener
          00
576 F68AB 3500 GETIDO  LC(6)  (=MSDI)+8     Max of 8 characters
          0000
577 F68B3 2D           P=     13            Set flag to indicate strip Cr
578 F68B5 A83          D=0    P
579 F68B8 A0F          D=D-1  P             D[13]="F"...strip returns
580 F68BB 7692 READRg  GOSUB  PUTE
581 F68BF 400          RTNC
582 F68C2 AF0          A=0    W             Preclear A[W]
583 F68C5 AC3          D=0    S             Clear D[S] (count)
584 F68C8 797E GETID1  GOSUB  GETX          Get a message
585 F68CC 487          GOC    GETID4        If carry, not data
586 F68CF AEA  GETID2  A=C    B
587 F68D2 814          ASRC
588 F68D5 814          ASRC                 Rotate into A[15:14]
589 F68D8 BF6          CSR    W             Shift next char into C[B]
590 F68DB F6           CSR    A             (at most GETX returns 6 nibs)
591 F68DD B47          D=D+1  S             Increment count
592 F68E0 0D           P=P-1
593 F68E2 5CE          GONC   GETID2        If no carry, more bytes
594            *
595            * If carry, P=15!
596            *
597 F68E5 308          LC(1)  8
```

```
598 F68E8 9C7          ?D<C   S
599 F68EB DD           GOYES  GETID1        Get more bytes
600 F68ED 20           P=     0             Now remove any Cr,Lf!
601 F68EF 31D0 GETID3  LC(2)  13            Check for <Cr>
602 F68F3 2D           P=     13
603 F68F5 90F          ?D#0   P             Strip flag set?
604 F68F8 50           GOYES  GETIDs        Yes...strip <Cr>s
605 F68FA AE2          C=0    B             No...don't strip <Cr>s
606 F68FD 2F   GETIDs  P=     15
607 F68FF 96A  GETID*  ?C=0   B             Stripping trailing chars?
608 F6902 A0           GOYES  GETID-        No...continue
609 F6904 966          ?A#C   B             Yes...match?
610 F6907 50           GOYES  GETID-        No...continue
611 F6909 A90          A=0    WP            Yes...clear anything after <Cr>
612 F690C 814  GETID-  ASRC
613 F690F 814          ASRC
614 F6912 0D           P=P-1
615 F6914 0D           P=P-1
616 F6916 58E          GONC   GETID*        If no carry, continue
617             *
618             * Now remove any trailing zero bytes (decrement count)
619             *
620 F6919 2D           P=     13            Check if strip flag set
621 F691B 90B          ?D=0   P
622 F691E 32           GOYES  GETID!        Not strip...exit
623 F6920 2F           P=     15
624 F6922 AC2          C=0    S             Preclear the count!
625 F6925 978          ?A=0   W             Is whole word zero?
626 F6928 61           GOYES  GETID%        Yes...set count=0!
627 F692A 90C  GETID^  ?A#0   P
628 F692D 70           GOYES  GETID#
629 F692F 0D           P=P-1
630 F6931 58F          GONC   GETID^        Go always
631             *-
632             *-
633 F6934 80FF GETID#  CPEX   15
634 F6938 81E          CSRB                 Now C[S] is # of characters-1
635 F693B B46          C=C+1  S             C[S] is # of characters
636 F693E AC7  GETID%  D=C    S             Reset count in D[S]
637 F6941 20   GETID!  P=     0             Reset P=0
638 F6943 03           RTNCC                Done...exit
639             *-
640             *-
641 F6945 890  GETID4  ?P=    =eABORT       Is this an abort or error?
642 F6948 00           RTNYES               Yes...tell caller
643 F694A 80FF         CPEX   15
644 F694E 8E00         GOSUBL =FRAME-       Check what it IS
          00
645 F6954 890          ?P=    =pSTATE       Current state?
646 F6957 B0           GOYES  GETID5        Yes...justify, return-carry clear
647 F6959 890          ?P=    =pEOT         EOT?
648 F695C 60           GOYES  GETID5        Yes...justify, return-carry clear
649             *
650             * NOT state or EOT...error!
651             *
```

```
652 F695E 66AD          GOTO   READI2        Unexpected frame
653             *-
654             *-
655 F6962 AC8  GETID5   C=D    S
656 F6965 80DF          P=C    15            P=count until justified!
657 F6969 890  GETID6   ?P=    0
658 F696C 38            GOYES  GETID3        Return, carry clear
659 F696E 810           ASLC
660 F6971 810           ASLC                 Shift one character
661 F6974 0D            P=P-1                Decrement character count
662 F6976 52F           GONC   GETID6        Go always
663             ********************************************************************
664             ********************************************************************
665             **
666             ** Name:       INITFL - Initialize a file on external device
667             **
668             ** Category:   FILUTL
669             **
670             ** Purpose:
671             **      Initialize an external file after creation
672             **
673             ** Entry:
674             **      R1[S] = Create code of the file
675             **      Tape is positioned at the start of the file data area
676             **      R2[A] is # of sectors in the file
677             **
678             ** Exit:
679             **      Carry clear:
680             **        The file will be filled with zeros or all FF's
681             **        Create code = 2 - filled with zeros
682             **        Otherwise - filled with all FF's
683             **      Carry set:
684             **        Error...P, C[0] are error code
685             **
686             ** Calls:      SENDIT
687             **
688             ** Uses:
689             **  Exclusive: A[W],C[W],D1,        FUNCR1[15:0],P
690             **  Inclusive: A[W],C[W],D1,ST[3:0],FUNCR1[15:0],P
691             **
692             ** Stk lvls:   2 (SENDIT)
693             **
694             ** History:
695             **
696             **    Date     Programmer              Modification
697             **  --------   ----------    ---------------------------------
698             **  09/21/83      NZ         Updated documentation
699             **  04/18/83      NZ         Modified entry conditions and
700             **                           rewrote routine to save code and
701             **                           fix several bugs
702             **  01/25/83      NZ         Updated documentation, changed
703             **                           code to cut 2B(hex) nibbles
704             **  10/01/82      SC         Wrote routine
705             **
706             ********************************************************************
```

```
707              ***************************************************************
708 F6979        =INITFL
709 F6979 1F00           D1=(5) =FUNCR1
          000
710 F6980 AF4            A=B    W              Get B[W] into A[W]
711 F6983 1517           DAT1=A W              Save B[W] in FUNCR1
712              *
713 F6987 112            A=R2                  Recall size in sectors
714 F698A F0             ASL    A
715              *
716              * If the file size can exceed 1M bytes, the following shift
717              * will produce erroneous results!!!!
718              *
719 F698C F0             ASL    A              Multiply by 256 bytes/sector
720              *
721 F698E AF1            B=0    W              Clear B[W] (pattern)
722 F6991 119            C=R1                  C(S)= CREATE CODE
723 F6994 80DF           P=C    15             Get CREATE code into P
724 F6998 892            ?P=    2              Create code=2?
725 F699B 50             GOYES  INIT10         Yes...pattern is zero
726 F699D A7D            B=B-1  W              No...pattern is "FFFFF"
727 F69A0 20    INIT10   P=     0              Reset P=0
728 F69A2 7E70           GOSUB  SENDIT         Now send the pattern!
729 F69A6 1537           A=DAT1 W              D1 unchanged by SENDIT!
730 F69AA AF8            B=A    W              Restore B[W]
731 F69AD 01             RTN                   Carry set if error, else clear
732              ***************************************************************
733              ***************************************************************
734              **
735              ** Name:       WRITIT - Write data from RAM to the mailbox
736              **
737              ** Category:   PILI/O
738              **
739              ** Purpose:
740              **      Output data to the Diamond, given a buffer of data in
741              **      RAM and a pointer (D1) to the buffer
742              **
743              ** Entry:
744              **      D0: Diamond mailbox
745              **      D1: Data buffer start
746              **      A[A]: Number of bytes of data to send from at D1
747              **      Loop is addressed, set up for this transfer
748              **      ST(=LoopOK) set if should abort on one ATTN, else clear
749              **
750              ** Exit:
751              **      Carry clear:
752              **        Transfer complete, D1 points past end of buffer,
753              **        A[A]="000FF", P unchanged from entry
754              **      Carry set: Error - P is the error number, A[A] is the
755              **        number of data bytes not sent (may be low by up to 3)
756              **        (If Attn key hit ONCE, then carry set, P=0)
757              **
758              ** Calls:      PUTX,PUTD,CK=ATN
759              **
760              ** Uses.......
```

```
                ** Exclusive: A[A],C[W],D1
                ** Inclusive: A[A],C[W],D1,ST[3:0]
                **
                ** Stk lvls:   1 (PUTX)(PUTD)(CK=ATN)
                **
                ** NOTE: this routine can be SLIGHTLY speeded up by calling
                **   PUTX one statement later (after the CPEX 15)...at the
                **   cost of setting P=0 unconditionally
                **
                ** History:
                **
                **    Date      Programmer            Modification
                **  --------    ----------    --------------------------------
                **  09/27/83       NZ         Installed fix of SR for Memory
                **                            Lost during OUTPUT and/or PRINT
                **                            (The bug was that WRITIT did not
                **                            check carry from PUTD, therefore
                **                            would return with carry clear,
                **                            but P= =eABORT or =ePIL)
                **  09/21/83       NZ         Updated documentation
                **  07/21/83       NZ         Added status for don't abort for
                **                            single ATTN hit
                **  03/15/83       NZ         Added P=0 if ATNFLG=F
                **  11/24/82       NZ         Added documentation
                **
                ****************************************************************
                ****************************************************************
F69AF 870   =WRITIT ?ST=1   =Attn
F69B2 E1            GOYES   WRITI1          ATTN hit at least once...check!
F69B4 132   WRITI0  ADOEX
F69B7 182           D0=D0-  3               See if three bytes to send
F69BA 4F1           GOC     WRITI2          No...transfer remaining bytes
F69BD 132           ADOEX
                *
                * Have three bytes to send
                *
F69C0 15F5          C=DAT1  6               Read three
F69C4 175           D1=D1+  6               Point to next
F69C7 7CC0          GOSUB   PUTX            Send them
F69CB 53E           GONC    WRITIT          Go unless Attn hit more than once
F69CE 02            RTNSC                   Error!
                *-
                *-
F69D0 7F20  WRITI1  GOSUB   CK=ATN
F69D4 4FD           GOC     WRITI0          Not ATTN key...continue

F69D7 572           GONC    P=0:SC          Go always (PACK 9/27/83 NZ)
                *
                *       P=      0           Attn key ONCE
                *       RTNSC               Attn key interrupt...exit!
                *-
                *-
F69DA 162   WRITI2  D0=D0+  3               Correct for over-subtracting
F69DD 132           ADOEX
F69E0 A6C   WRITI3  A=A-1   B               If carry, than done
```

```
816 F69E3 4D6          GOC    WRITI4        Done!
817 F69E6 14F          C=DAT1 B             Read it...
818 F69E9 171          D1=D1+ 2             Next byte...
819 F69EC 7351         GOSUB  PUTD          Send it!
820             *
821             * Following RTNC is bug fix on 9/27/83 by NZ
822             *
823 F69F0 400          RTNC                 Error...set carry
824             *
825 F69F3 860          ?ST=0  =Attn
826 F69F6 AE           GOYES  WRITI3        Loop back if not interrupt
827 F69F8 7700         GOSUB  CK=ATN
828 F69FC 43E          GOC    WRITI3        Loop back if not interrupt
829 F69FF 20   P=0:SC  P=     0             Attn key ONCE
830 F6A01 02           RTNSC                Attn key interrupt...exit!
831             *-
832             *-
833             *
834             * Moved to location below by NZ on 9/27/83 as part of bug fix
835             *
836             *WRITI4 RTNCC                Done...return with carry clear!
837             *-
838             *-
839             *
840             * CK=ATN will return with carry set if OK to continue, clear
841             * if time to abort transmission
842             *
843 F6A03 860  =CK=ATN ?ST=0  =LoopOK        Should I check ATNFLG?
844 F6A06 00           RTNYES               No...say OK
845 F6A08 136  =CK=ATn CDOEX                Save D0 in C[A]
846 F6A0B 1B00         D0=(5) =ATNFLG
        000
847 F6A12 1564         C=DAT0 S
848 F6A16 134          D0=C                 Restore D0
849 F6A19 A4E          C=C-1  S             If carry, ATNFLG was zero
850 F6A1C 01           RTN
851             ****************************************************************
852             ****************************************************************
853             **
854             ** Name:       SENDIT - Send a 1 or 2 char sequence from B[W]
855             ** Name:       SENDI+ - Find mailbox, send a sequence of chars
856             **
857             ** Category:   PILI/O
858             **
859             ** Purpose:
860             **       Send a sequence of 1 or 2 characters (in B[7:0])
861             **       Number of characters to send in A[A]
862             **
863             ** Entry:
864             **       A[A]=count of characters
865             **       B[7:0]=sequence (B[B]=first char, B[3:2]=second char,
866             **         B[5:4]=first char, B[7:6]=second char)
867             **       D0 points to mailbox
868             **       ST(=LoopOK) set if abort on 1 ATTN, else clear
869             **
```

```
                ** Exit:
                **      Carry set if Attn or error, else clear
                **      If carry set and P=0, then ATTN key hit ONCE
                **
                ** Calls:      PUTX,PUTD,CK=ATN  (SENDI+ also calls GETMBX)
                **
                ** Uses.......
                **  Exclusive: A[A],C[W]
                **  Inclusive: A[A],C[W],ST[3:0]
                **
                ** Stk lvls:   1 (PUTX)(PUTD)(CK=ATN)(GETMBX)
                **
                ** NOTE: This routine can be speeded up SLIGHTLY...see WRITIT
                ** documentation)
                **
                ** History:
                **
                **    Date      Programmer            Modification
                **   --------   ----------   --------------------------------
                **   09/27/83      NZ        Packed code (needed for WRITIT fix)
                **   09/21/83      NZ        Updated documentation
                **   03/15/83      NZ        Added P=0 for Attn key ONCE
                **   11/24/82      NZ        Added documentation
                **
                ****************************************************************
                ****************************************************************
F6A1E 8E00 =SENDI+ GOSUBL =GETMBX
      00
F6A24 870  =SENDIT ?ST=1  =Attn          Check if immediate exit
F6A27 C2           GOYES  SENDI1
F6A29 132  SENDIO  ADOEX
F6A2C 185          DO=DO- 6
F6A2F 4D2          GOC    SENDI2         Less than 6 left
F6A32 132          ADOEX
F6A35 AF9          C=B    W
F6A38 7B50         GOSUB  PUTX           Send first 3 chars
F6A3C 400          RTNC                  Attn
F6A3F AF9          C=B    W
F6A42 BF6          CSR    W
F6A45 BF6          CSR    W
F6A48 7B40         GOSUB  PUTX           Send next 3 chars
F6A4C 57D          GONC   SENDIT         Loop back!
F6A4F 02           RTNSC                 Error!
                *-
                *-
F6A51 03   WRITI4  RTNCC                 Moved here 9/27/83 by NZ
                *-
                *-
F6A53 7CAF SENDI1  GOSUB  CK=ATN
F6A57 41D          GOC    SENDIO         Not ATTN key...continue
F6A5A 54A  P=0:sc  GONC   P=0:SC         Packed 9/27/83 by NZ
                *
                *       P=     0         Attn key ONCE
                *       RTNSC            Attn key interrupt...exit!
                *-
```

```
924             *-
925 F6A5D 165   SENDI2  DO=DO+ 6
926 F6A60 132           ADOEX
927 F6A63 A6C   SENDI3  A=A-1  B
928 F6A66 4E2           GOC    SENDI4        Done if carry
929 F6A69 AE9           C=B    B
930 F6A6C 73D0          GOSUB  PUTD          Send first byte
931 F6A70 400           RTNC                 Attn
932 F6A73 A6C           A=A-1  B
933 F6A76 4E1           GOC    SENDI4        Done if carry
934 F6A79 D9            C=B    A
935 F6A7B F6            CSR    A
936 F6A7D F6            CSR    A
937 F6A7F 70C0          GOSUB  PUTD          Send second byte
938 F6A83 400           RTNC
939 F6A86 860           ?ST=0  =Attn
940 F6A89 AD            GOYES  SENDI3        Loop back if not interrupt
941 F6A8B 747F          GOSUB  CK=ATN
942 F6A8F 43D           GOC    SENDI3        Not ATTN key...continue
943 F6A92 57C           GONC   P=0:sc        Packed 9/27/83 by NZ
944             *
945             *       P=     0             Attn key ONCE
946             *       RTNSC                Attn key interrupt...exit!
947             *-
948             *-
949 F6A95 03    SENDI4  RTNCC                Done!
950             ********************************************************************
951             ********************************************************************
952             **
953             ** Name:       PUTX - Send 3 bytes of data from C[5:0] to loop
954             **
955             ** Category:   PILI/O
956             **
957             ** Purpose:
958             **      Output three bytes from C[5:0] to PIL
959             **
960             ** Entry:
961             **      C[5:0] is the three data bytes (C[B] is first byte)
962             **      DO: HPIL mailbox
963             **
964             ** Exit:
965             **      Carry clear: done
966             **      Carry set: error (P is error #)
967             **
968             ** Calls:      None
969             **
970             ** Uses.......
971             **  Inclusive: C[W],ST[3:0]
972             **
973             ** Stk lvls:   0
974             **
975             ** History:
976             **
977             **    Date     Programmer              Modification
978             **  --------   ----------    ---------------------------------
```

```
 979            **  03/15/83      NZ          Removed check for Attn at PUTX5
 980            **                            to insure Error is always checked
 981            **  03/07/83      NZ          Added flag to ignore error bit
 982            **  03/04/83      NZ          Reordered code to check sERROR
 983            **                            ONLY if ATNFLG is non-zero
 984            **  03/02/83      NZ          Added check for sERROR if Attn is
 985            **                            set
 986            **  11/24/82      NZ          Added documentation
 987            **
 988            ****************************************************************
 989            ****************************************************************
 990 F6A97 80FF =PUTX   CPEX   15             Save P in C[S]
 991 F6A9B 26           P=     6
 992 F6A9D 3181         LCHEX  18             Long transfer bits...
 993 F6AA1 166  PUTXx   D0=D0+ oOUTHS
 994 F6AA4 80DF         P=C    15             Restore P
 995 F6AA8 870          ?ST=1  =Attn
 996 F6AAB D1           GOYES  PUTX3          Check for immediate abort!
 997 F6AAD 0B   PUTX1   CSTEX
 998 F6AAF 15E0         C=DAT0 1              Read the handshake
 999 F6AB3 0B           CSTEX
1000 F6AB5 871          ?ST=1  NRD            NRD?
1001 F6AB8 01           GOYES  PUTX3          Yes...wait!
1002 F6ABA 870          ?ST#0  MAV
1003 F6ABD B0           GOYES  PUTX3
1004 F6ABF 186  PUTEx   D0=D0- oOUTHS
1005            *
1006            * Ready to send it now (coast is clear)
1007            *
1008 F6AC2 15C7         DAT0=C 8
1009 F6AC6 03           RTNCC
1010            *-
1011            *-
1012 F6AC8 850  PUTX3   ST=1   sPUTX          Flag for return routine
1013            *
1014            * If here, not ready yet...check for ATTN
1015            *
1016 F6ACB 851  PUTX4   ST=1   sCHKER         D0 check error bit
1017 F6ACE      PUTX5
1018            *
1019            * Check =ATNFLG in RAM...
1020            *
1021            * Save the message in C[12:5] to check ATNFLG
1022            *
1023 F6ACE BF2          CSL    W
1024 F6AD1 BF2          CSL    W
1025 F6AD4 BF2          CSL    W
1026 F6AD7 BF2          CSL    W
1027 F6ADA BF2          CSL    W              Now message in C[12:5]
1028 F6ADD 136          CDOEX
1029 F6AE0 1B00         D0=(5) =ATNFLG
          000
1030 F6AE7 1564         C=DAT0 S
1031 F6AEB 134          D0=C                  Restore D0
1032            *
```

```
1033 F6AEE 161          D0=D0+ (oINST)-(oOUTHS)
1034 F6AF1 15E0         C=DAT0 1            Read the status nibble &check err
1035 F6AF5 181          D0=D0- (oINST)-(oOUTHS)
1036 F6AF8 0B           CSTEX
1037 F6AFA 870          ?ST=1  =sERROR
1038 F6AFD 20           GOYES  PUTx0
1039 F6AFF 0B   PUTx0   CSTEX               Carry SET if error bit set
1040                    *
1041 F6B01 BF6          CSR    W            Restore message to C[7:0]
1042 F6B04 BF6          CSR    W
1043 F6B07 BF6          CSR    W
1044 F6B0A BF6          CSR    W
1045 F6B0D BF6          CSR    W            Now ATNFLG is in C[8]
1046 F6B10 80FA         CPEX   10
1047 F6B14 570          GONC   PUTx1        No carry...sERROR clear
1048 F6B17 871          ?ST=1  sCHKER       Check error bit?
1049 F6B1A C0           GOYES  PUTx.        Yes...error!
1050                    *
1051                    * If sCHKER=0, then ignore the error bit
1052                    *
1053 F6B1C 890  PUTx1   ?P=    0            ATTN key?
1054 F6B1F E0           GOYES  PUTx3        No...continue
1055 F6B21 0C           P=P+1
1056 F6B23 490          GOC    PUTx3        Attn key, hit only ONCE
1057                    *
1058                    * ATTN key hit!
1059                    *
1060 F6B26 186  PUTx.   D0=D0- oOUTHS
1061 F6B29 20           P=     =eABORT      Aborted by ATTN key
1062 F6B2B 02           RTNSC
1063                    *-
1064                    *-
1065 F6B2D 80FA PUTx3   CPEX   10           Restore P!
1066 F6B31 861  PUTX6   ?ST=0  sCHKER       Is this PUTN?
1067 F6B34 B0           GOYES  PUTX7        Yes...loop
1068 F6B36 860          ?ST#1  sPUTX
1069 F6B39 C2           GOYES  PUTE1        Loop again
1070 F6B3B 617F         GOTO   PUTX1        (Out of range)
1071                    *-
1072                    *-
1073 F6B3F 6650 PUTX7   GOTO   PUTN1        Continue with PUTN
1074                    ****************************************************************
1075                    ****************************************************************
1076                    **
1077                    ** Name:       PUTD - Put a single data byte on the loop
1078                    **
1079                    ** Category:   PILI/O
1080                    **
1081                    ** Purpose:
1082                    **      Send a single data byte on the loop (Check NRD first)
1083                    **
1084                    ** Entry:
1085                    **      C[B] contains the data byte
1086                    **      D0 points to the HPIL mailbox
1087                    **
```

```
1088                ** Exit:
1089                **      Handshake nibble in ST[3:0]
1090                **      Carry set if error, clear if OK
1091                **
1092                ** Calls:      None
1093                **
1094                ** Uses.......
1095                **  Inclusive: C[W],ST[3:0]
1096                **
1097                ** Stk lvls:   0
1098                **
1099                ** History:
1100                **
1101                **    Date      Programmer             Modification
1102                **  --------    ----------    --------------------------------
1103                **  02/18/83       NZ         Changed to share code with PUTX
1104                **  11/24/82       NZ         Added documentation
1105                **
1106                ********************************************************************
1107                ********************************************************************
1108 F6B43 80FF =PUTD   CPEX   15
1109 F6B47 22         P=     2
1110 F6B49 3500       LC(6)  #140000       This is a single data frame
           0041
1111 F6B51 6F4F       GOTO   PUTXx         Continue with common code in PUTX
1112                ********************************************************************
1113                ********************************************************************
1114                **
1115                ** Name:       PUTE - Put extended message (6 nibbles)
1116                ** Name:       PUTEX - Put extended message (6 nibs + 2 hs)
1117                **
1118                ** Category:   PILI/O
1119                **
1120                ** Purpose:
1121                **      PUTE:Put extended mailbox message (given full 6 nibs)
1122                **      PUTEX:Put a full message, INCLUDING HANDSHAKE!!!!
1123                **
1124                ** Entry:
1125                **      PUTE: C[5:0] is message
1126                **      PUTEX: C[7:0] is message
1127                **      D0 points to the mailbox
1128                **
1129                ** Exit:
1130                **      Carry clear: OK (P=0 for PUTX)
1131                **      Carry set: error (P=error #)
1132                **
1133                ** Calls:      None
1134                **
1135                ** Uses.......
1136                **  Inclusive: C,ST[3:0] (PUTE sets P=0)
1137                **
1138                ** Stk lvls:   0
1139                **
1140                ** History:
1141                **
```

```
1142                    **    Date     Programmer            Modification
1143                    **  --------   ----------   ---------------------------------
1144                    **  02/18/83      NZ        Packed by sharing code with PUTX
1145                    **  11/24/82      NZ        Added documentation
1146                    **
1147                    ****************************************************************
1148                    ****************************************************************
1149 F6B55 26          =PUTE   P=     6
1150 F6B57 3101                LCHEX  10
1151 F6B5B 20                  P=     0
1152 F6B5D             =PUTEX
1153 F6B5D 166                 DO=DO+ oOUTHS
1154 F6B60 870                 ?ST=1  =Attn
1155 F6B63 31                  GOYES  PUTE2          Check for immediate abort
1156 F6B65 0B          PUTE1   CSTEX
1157 F6B67 15E0                 C=DATO 1              Read handshake nibble
1158 F6B6B 0B                   CSTEX
1159 F6B6D 870                  ?ST#0  MAV
1160 F6B70 60                   GOYES  PUTE2
1161 F6B72 6C4F PUTEx.  GOTO    PUTEx                 Can be GONC if it will reach!
1162                    *-
1163                    *-
1164                    *
1165                    * Looping...check ATTN flag
1166                    *
1167 F6B76 840  PUTE2   ST=0    sPUTX
1168 F6B79 615F                GOTO   PUTX4          Check for ATTN flag, return:PUTE1
1169                    ****************************************************************
1170                    ****************************************************************
1171                    **
1172                    ** Name:       PUTEN - Put message in C[5:0], don't check error
1173                    ** Name:       PUTCN - Put message in C[3:0], don't check error
1174                    ** Name:       PUTC+N - Put message in C[B], don't check error
1175                    **
1176                    ** Category:   PILI/O
1177                    **
1178                    ** Purpose:
1179                    **       Put a message without checking for the Diamond error
1180                    **       bit (otherwise same as PUTE)
1181                    **
1182                    ** Entry:
1183                    **       DO points to the HPIL mailbox
1184                    **
1185                    **       PUTEN: Message in C[5:0]
1186                    **       PUTCN: Message in C[3:0]
1187                    **       PUTC+N: Message in C[B]
1188                    **
1189                    ** Exit:
1190                    **       Carry clear:
1191                    **         Handshake nibble in ST[3:0]
1192                    **       Carry set:
1193                    **         P=error #
1194                    **
1195                    ** Calls:      None
1196                    **
```

```
1197            ** Uses.......
1198            **  Exclusive: C[W]
1199            **  Inclusive: C[W],ST[3:0]
1200            **
1201            ** Stk lvls:   0
1202            **
1203            ** History:
1204            **
1205            **    Date      Programmer              Modification
1206            **  --------    ----------    --------------------------------
1207            **  09/21/83      NZ          Added documentation
1208            **
1209            ************************************************************
1210            ************************************************************
1211 F6B7D F2   =PUTC+N CSL     A             PUTC+ except don't check error
1212 F6B7F F2           CSL     A
1213 F6B81 F2   =PUTCN  CSL     A             PUTC except don't check error
1214 F6B83 BF2          CSL     W
1215 F6B86 26   =PUTEN  P=      6             PUTE except don't check error
1216 F6B88 3101         LCHEX   10
1217 F6B8C 20           P=      0
1218 F6B8E 166          DO=DO+  oOUTHS
1219 F6B91 870          ?ST=1   =Attn
1220 F6B94 21           GOYES   PUTN2
1221 F6B96 0B   PUTN1   CSTEX
1222 F6B98 15E0         C=DATO  1             Read handshake
1223 F6B9C 0B           CSTEX
1224 F6B9E 870          ?ST#0   MAV           Message available?
1225 F6BA1 50           GOYES   PUTN2         No...wait loop
1226 F6BA3 5EC          GONC    PUTEx.        Go always...jump to finish
1227            *-
1228            *-
1229 F6BA6 841  PUTN2   ST=0    sCHKER        Don't check error!
1230 F6BA9 642F         GOTO    PUTX5
1231            ************************************************************
1232            ************************************************************
1233            **
1234            ** Name:       PUTC+ - Put a command (1 byte) to the mailbox
1235            ** Name:       PUTC - Put a command (2 bytes) to the mailbox
1236            **
1237            ** Category:   PILI/O
1238            **
1239            ** Purpose:
1240            **      Put a command (1 or 2 bytes) to the mailbox
1241            **
1242            ** Entry:
1243            **      DO points to the HPIL mailbox
1244            **      PUTC+: C[B] contains the command to send (1 byte)
1245            **      PUTC: C[3:0] contains the command to send (2 bytes)
1246            **
1247            ** Exit:
1248            **      Same as PUTE
1249            **
1250            ** Calls:      None
1251            **
```

```
1252                    ** Uses.......
1253                    **  Inclusive: C[W],ST[3:0],P
1254                    **
1255                    ** Stk lvls:   0
1256                    **
1257                    ** History:
1258                    **
1259                    **    Date      Programmer              Modification
1260                    **  --------    ----------    --------------------------------
1261                    **  09/21/83       NZ         Updated documentation
1262                    **  11/24/82       NZ         Added documentation
1263                    **
1264                    ****************************************************************
1265                    ****************************************************************
1266 F6BAD F2         =PUTC+  CSL     A
1267 F6BAF F2                 CSL     A
1268 F6BB1 F2         =PUTC   CSL     A
1269 F6BB3 BF2                CSL     W
1270 F6BB6 6E9F               GOTO    PUTE            Continue as if PUTE
1271                    ****************************************************************
1272                    ****************************************************************
1273                    **
1274                    ** Name:       DDT,DDL - Send a Device Dependent Command
1275                    **
1276                    ** Category:   PILI/O
1277                    **
1278                    ** Purpose:
1279                    **      Send a DDL/DDT as determined by P (these routines are
1280                    **      only good for DDL/DDT 0-15)
1281                    **
1282                    ** Entry:
1283                    **      P contains the DDL/DDT number desired
1284                    **      Loop is set up
1285                    **      DO @ mailbox
1286                    **
1287                    ** Exit:
1288                    **      Same as PUTE
1289                    **
1290                    ** Calls:      None
1291                    **
1292                    ** Uses.......
1293                    **  Inclusive: C[W],ST[3:0],P
1294                    **
1295                    ** Stk lvls:   0
1296                    **
1297                    ** History:
1298                    **
1299                    **    Date      Programmer              Modification
1300                    **  --------    ----------    --------------------------------
1301                    **  11/24/82       NZ         Added documentation
1302                    **
1303                    ****************************************************************
1304                    ****************************************************************
1305 F6BBA 80F0   =DDL    CPEX    0
1306 F6BBE 21             P=      1
```

```
1307 F6BC0 3200          LC(3)  (=mCMD3)+#A   DDL
           0
1308 F6BC5 6BEF          GOTO   PUTC
1309             *-
1310             *-
1311 F6BC9 80F0 =DDT     CPEX   0
1312 F6BCD 21            P=     1
1313 F6BCF 3200          LC(3)  (=mCMD3)+#C   DDT
           0
1314 F6BD4 6CDF          GOTO   PUTC
1315 F6BD8               END
```

```
 ATNFLG  Ext                  -   252   846  1029
 Attn    Ext                  -   246   788   825   897   939   995  1154  1219
 BADRD1  Abs 1009454 #F672E   -   139   130
=CHECKD  Abs 1009764 #F6864   -   498
=CHKEND  Abs 1009793 #F6881   -   508
=CK=ATN  Abs 1010179 #F6A03   -   843   804   827   917   941
=CK=ATn  Abs 1010184 #F6A08   -   845
=DDL     Abs 1010618 #F6BBA   - 1305
=DDT     Abs 1010633 #F6BC9   - 1311
 FRAME+  Ext                  -   441   498   508
 FRAME-  Ext                  -   128   644
 FUNCR1  Ext                  -   709
=GET     Abs 1009638 #F67E6   -   318   496   506
 GET1    Abs 1009641 #F67E9   -   320   267
 GET2    Abs 1009654 #F67F6   -   328   315
 GET9    Abs 1009669 #F6805   -   338   324
=GETD    Abs 1009757 #F685D   -   496
 GETD1   Abs 1009775 #F686F   -   501   500   510   511
=GETEND  Abs 1009786 #F687A   -   506
 GETER0  Abs 1009708 #F682C   -   437   432
 GETER3  Abs 1009749 #F6855   -   453   447
=GETERR  Abs 1009702 #F6826   -   435
=GETHS2  Abs 1009676 #F680C   -   371
=GETID   Abs 1009827 #F68A3   -   574
 GETID!  Abs 1009985 #F6941   -   637   622
 GETID#  Abs 1009972 #F6934   -   633   628
 GETID%  Abs 1009982 #F693E   -   636   626
 GETID*  Abs 1009919 #F68FF   -   607   616
=GETID+  Abs 1009807 #F688F   -   564
 GETID-  Abs 1009932 #F690C   -   612   608   610
 GETID0  Abs 1009835 #F68AB   -   576   565
 GETID1  Abs 1009864 #F68C8   -   584   599
 GETID2  Abs 1009871 #F68CF   -   586   593
 GETID3  Abs 1009903 #F68EF   -   601   658
 GETID4  Abs 1009989 #F6945   -   641   585
 GETID5  Abs 1010018 #F6962   -   655   646   648
 GETID6  Abs 1010025 #F6969   -   657   662
 GETID^  Abs 1009962 #F692A   -   627   630
 GETIDs  Abs 1009917 #F68FD   -   606   604
 GETMBX  Ext                  -   896
 GETNO   Abs 1009619 #F67D3   -   309   265
=GETNE   Abs 1009616 #F67D0   -   308   439
=GETST   Abs 1009692 #F681C   -   430
=GETST-  Abs 1009715 #F6833   -   439   443
 GETST2  Abs 1009749 #F6855   -   452   140
=GETX    Abs 1009477 #F6745   -   198   105   584
 GETX.   Abs 1009602 #F67C2   -   264   247   256   258
 GETX1   Abs 1009480 #F6748   -   199   268
 GETX2   Abs 1009493 #F6755   -   205
 GETX3   Abs 1009516 #F676C   -   215   210
 GETX4   Abs 1009525 #F6775   -   222   221
 GETXE   Abs 1009529 #F6779   -   227   203
 GETXNE  Abs 1009535 #F677F   -   229   314
 GETx.   Abs 1009532 #F677C   -   228   339
 GETx..  Abs 1009557 #F6795   -   241   240
```

```
 GETx.N  Abs 1009562 #F679A -   246   234
 GETxE   Abs 1009595 #F67BB -   259   242
 INIT10  Abs 1010080 #F69A0 -   727   725
=INITFL  Abs 1010041 #F6979 -   708
 LoopOK  Ext                   843
 MAV     Abs       0 #00000 -    18   202   313   323  1002  1159  1224
 NRD     Abs       1 #00001 -    19  1000
 P=0:SC  Abs 1010175 #F69FF -   829   807   919
 P=0:sc  Abs 1010266 #F6A5A -   919   943
=PUTC    Abs 1010609 #F6BB1 -  1268  1308  1314
=PUTC+   Abs 1010605 #F6BAD -  1266
=PUTC+N  Abs 1010557 #F6B7D -  1211   437
=PUTCN   Abs 1010561 #F6B81 -  1213
=PUTD    Abs 1010499 #F6B43 -  1108   819   930   937
=PUTE    Abs 1010517 #F6B55 -  1149    98   580  1270
 PUTE1   Abs 1010533 #F6B65 -  1156  1069
 PUTE2   Abs 1010550 #F6B76 -  1167  1155  1160
=PUTEN   Abs 1010566 #F6B86 -  1215
=PUTEX   Abs 1010525 #F6B5D -  1152
 PUTEx   Abs 1010367 #F6ABF -  1004  1161
 PUTEx.  Abs 1010546 #F6B72 -  1161  1226
 PUTN1   Abs 1010582 #F6B96 -  1221  1073
 PUTN2   Abs 1010598 #F6BA6 -  1229  1220  1225
=PUTX    Abs 1010327 #F6A97 -   990   799   904   909
 PUTX1   Abs 1010349 #F6AAD -   997  1070
 PUTX3   Abs 1010376 #F6AC8 -  1012   996  1001  1003
 PUTX4   Abs 1010379 #F6ACB -  1016  1168
 PUTX5   Abs 1010382 #F6ACE -  1017  1230
 PUTX6   Abs 1010481 #F6B31 -  1066
 PUTX7   Abs 1010495 #F6B3F -  1073  1067
 PUTXx   Abs 1010337 #F6AA1 -   993  1111
 PUTx.   Abs 1010470 #F6B26 -  1060  1049
 PUTx0   Abs 1010431 #F6AFF -  1039  1038
 PUTx1   Abs 1010460 #F6B1C -  1053  1047
 PUTx3   Abs 1010477 #F6B2D -  1065  1054  1056
 READER  Abs 1009422 #F670E -   126   106
 READI2  Abs 1009413 #F6705 -   120   113   136   503   652
=READI3  Abs 1009462 #F6736 -   143   108
 READI9  Abs 1009475 #F6743 -   154   104
=READIT  Abs 1009374 #F66DE -   103   116   147
=READRG  Abs 1009818 #F689A -   569
 READRg  Abs 1009851 #F68BB -   580   571
 READS+  Abs 1009367 #F66D7 -    98   133   135
=READSU  Abs 1009362 #F66D2 -    96
=SENDI+  Abs 1010206 #F6A1E -   896
 SENDI0  Abs 1010217 #F6A29 -   899   918
 SENDI1  Abs 1010259 #F6A53 -   917   898
 SENDI2  Abs 1010269 #F6A5D -   925   901
 SENDI3  Abs 1010275 #F6A63 -   927   940   942
 SENDI4  Abs 1010325 #F6A95 -   949   928   933
=SENDIT  Abs 1010212 #F6A24 -   897   728   910
 WRITI0  Abs 1010100 #F69B4 -   790   805
 WRITI1  Abs 1010128 #F69D0 -   804   789
 WRITI2  Abs 1010138 #F69DA -   813   792
 WRITI3  Abs 1010144 #F69E0 -   815   826   828
```

```
 WRITI4  Abs 1010257 #F6A51 -   914   816
=WRITIT  Abs 1010095 #F69AF -   788   800
 YTML    Ext                -   564
 YTMLL   Ext                -   575
 eABORT  Ext                -   126   260   641  1061
 ePIL    Ext                -   122   454
 eUNEXP  Ext                -   121
 mCMD3   Ext                -  1307  1313
 mERSTS  Ext                -   436
 mSDI    Ext                -   576
 mSTATS  Ext                -   431
=oINHS   Abs       8 #00008 -    21   198   205   235   237   259   308   318
                              328
=oINST   Abs       9 #00009 -    22   205   208   235   237   328   331   372
                              374  1033  1035
=oOUTHS  Abs       7 #00007 -    17   993  1004  1033  1035  1060  1153  1218
=oOUTST  Abs       6 #00006 -    16
 pDATA   Ext                -   499
 pEOT    Ext                -   132   509   647
 pSTATE  Ext                -   129   442   645
 pTERM   Ext                -   134
 sCHKER  Abs       1 #00001 -    28   228   233   264   312  1016  1048  1066
                             1229
 sERROR  Ext                -   239  1037
 sGETX   Abs       0 #00000 -    27   227   266   338
 sPUTX   Abs       0 #00000 -    26  1012  1068  1167
```

Input Parameters

  Source file name is NZ&IOR::MS

  Listing file name is NZ/IOR:TI:ML::-1

  Object file name is NZ%IOR:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
                *
                *       N   N  ZZZZZ   &     FFFFF  RRRR     A
                *       N   N      Z  & &    F      R   R   A A
                *       NN  N     Z   & &    F      R   R  A   A
                *       N N N    Z     &     FFFF   RRRR   A   A
                *       N  NN   Z     & & &  F      R R    AAAAA
                *       N   N  Z      &  &   F      R  R   A   A
                *       N   N  ZZZZZ   && &  F      R   R  A   A
                *
                *
                        TITLE  PIL Frame Routines<831012.1534>
F6BD8           ABS    #F6BD8        TI%HP6 address (fixed)
                ****************************************************************
                ****************************************************************
                **
                ** Name:       FRAMEE - Encode an HPIL frame from its mnemonic
                **
                ** Category:   PILUTL
                **
                ** Purpose:
                **      HPIL frame encode (given the ASCII for the frame and a
                **      value, produce the appropriate 11-bit frame)
                **
                ** Entry:
                **      C[S] is length of ASCII character string
                **      C[S:0] is the ASCII character string
                **      A[B] is the value included with the frame (if none, 0)
                **
                ** Exit:
                **      P=0
                **      Carry clear: C[X] is the frame value
                **                   B[B] is the mask value for the frame
                **                   C[S] is WP length of name
                **      Carry set: Error...not found
                **
                ** Calls:      None
                **
                ** Uses.......
                **  Inclusive: B[W],C[W],P
                **
                ** Stk lvls:   1 (Internal push)
                **
                ** History:
                **
                **    Date     Programmer              Modification
                **  --------   ----------    ---------------------------------
                **  09/26/83      NZ         Updated documentation
                **
                ****************************************************************
                ****************************************************************
F6BD8           =FRAMEE
                *
                * C[5:0] is the ASCII frame value now
                *
F6BD8 AF5               B=C    W            Copy the ASCII to B[W] for now
```

```
 56 F6BDB 7000          GOSUB  FRAMSb
 57 F6BDF 07    FRAMSb  C=RSTK                 Now C[A] has the address of FRAMSb
 58 F6BE1 136           CDOEX                  ...now in DO.
 59 F6BE4 06            RSTK=C                 Save DO on the stack...
 60             *
 61             * Swap value of frame # into DO, address of FRAMSb into A[A]...
 62             *
 63 F6BE6 132           ADOEX
 64 F6BE9 20            P=     0
 65 F6BEB 346A          LC(5)  (FRAMET)-(FRAMSb)+#4  Offset to table + #4
          000
 66 F6BF2 CA            A=A+C  A
 67 F6BF4 132           ADOEX                  Restore A[A], set DO to table+4
 68             *
 69             * Now DO points to the frame table, A is the frame #
 70             *
 71 F6BF7 1567  FRAME1  C=DATO W               Read the ASCII for the current frame
 72 F6BFB 80D0          P=C    0
 73             *
 74             * Now P is the frame length
 75             *
 76 F6BFF BF6           CSR    W               Shift off the length nibble
 77 F6C02 890           ?P=    0               If length=0, not found...
 78 F6C05 27            GOYES  FRAME9          Not found!
 79             *
 80             * Now have a valid ASCII string in C[5:0]
 81             *
 82 F6C07 911           ?B=C   WP
 83 F6C0A 11            GOYES  FRAME2          Found a match!
 84             *
 85             * This does not match...try again!
 86             *
 87 F6C0C 164           DO=DO+ 5               Skip frame bits and text length
 88 F6C0F 136           CDOEX
 89 F6C12 809           C+P+1                  Add text length to DO
 90 F6C15 136           CDOEX
 91 F6C18 5ED           GONC   FRAME1          Go always
 92             *-
 93             *-
 94 F6C1B       FRAME2
 95             *
 96             * When here, had an ASCII match!
 97             *
 98 F6C1B 80FF          CPEX   15              Save length (P) in C[S]
 99             *
100             * Preset B[X] to #FFF (For mask)
101             *
102 F6C1F D1            B=0    A
103 F6C21 CD            B=B-1  A               B[X]=#FFF
104 F6C23 183           DO=DO- 4               Point to start of entry...
105 F6C26 15E3          C=DATO 4               ...and read the frame value+info
106 F6C2A 23            P=     3               Point to the status nibble
107 F6C2C A06           C=C+C  P               Is this a command bits only frame?
108 F6C2F 5B0           GONC   FRAME3          No...continue
109             *
```

```
110             * Copy the low 8 bits from A[B]!
111             *
112 F6C32 AE6           C=A    B
113 F6C35 AE1           B=0    B               Clear low 8 bits of mask
114 F6C38 473           GOC    FRAME8          Exit, carry cleared by FRAME8
115             *-
116             *-
117 F6C3B A06   FRAME3  C=C+C  P               Is this a low 5 bits only?
118 F6C3E 532           GONC   FRAME4          No...continue
119             *
120             * Need to copy the low 5 bits of A[B] into C[B]
121             *
122 F6C41 D5            B=C    A               Temporary storage!
123 F6C43 20            P=     0
124 F6C45 320E          LCHEX  FE0             Mask for low 5 bits
          F
125 F6C4A 0EF1          B=B&C  A               Now B[X] is the high bits of frame
126 F6C4E FE            C=-C-1 A               One's complement of C[X]
127 F6C50 0EF2          C=A&C  A               Now C[X] is the low bits of frame
128 F6C54 0EF9          B=C!B  A               Now B[X] is the full frame
129 F6C58 320E          LCHEX  FE0             Mask value
          F
130 F6C5D DD            BCEX   A               Mask in B[X], Frame in C[X]
131             *
132             * C=-C-1 above cleared the carry unconditionally
133             *
134 F6C5F 501           GONC   FRAME8          Go always-exit, clear carry
135             *-
136             *-
137 F6C62 A06   FRAME4  C=C+C  P               Low 4 bits?
138 F6C65 5A0           GONC   FRAME8          No...full frame!
139             *
140             * This is a low 4 bits case...
141             *
142 F6C68 20            P=     0
143 F6C6A A86           C=A    P
144 F6C6D A81           B=0    P               Clear low 4 bits of mask
145             *
146             * Now C[X] is the frame...clear carry, then restore data
147             *
148 F6C70       FRAME8
149 F6C70 BED           B=-B-1 B               Set B[B] to mask for frame
150 F6C73 21            P=     1
151 F6C75 0D            P=P-1                  Now P=0, carry is clear
152             *
153             * Now restore the data
154             *
155 F6C77 136   FRAME9  CDOEX                  These instructions don't alter carry
156 F6C7A 07            C=RSTK                 Restore DO value
157 F6C7C 136           CDOEX
158 F6C7F 01            RTN
159             ****************************************************************
160             ****************************************************************
161             **
162             ** Name:       FRAMET - Frame table format
```

```
163             **
164             ** Category:   LOCAL
165             **
166             ** Purpose:
167             **      Table of entries for frame encoding/decoding
168             **      (ASCII vs frame value)
169             **
170             ** Detail:
171             **      Format of entries as seen in RAM:
172             **
173             **      Length (nibbles)     Definition
174             **      ----------------     --------------------------------
175             **            3              Frame value (least sig nib first)
176             **            1              Control bits:
177             **                             8: Command bits only
178             **                             4: High 6 bits only
179             **                             2: High 7 bits only
180             **                             0: All bits meaningful
181             **            1              Text length (WP value)
182             **         (Length+1)        Text of frame
183             **
184             **      As read into the A register:
185             **        A[<--Text-->,<--length-->,<--control-->,<--frame-->]
186             **      nib:15.......5,4.........4,3.........3,2.........0
187             **
188             ********************************************************************
189             ********************************************************************
190 F6C81       =FRAMET
191             *
192             Comand  EQU    8
193             High6   EQU    4
194             High7   EQU    2
195             Allbit  EQU    0
196             *
197             * Frame classes (no subdivisions)
198             *
199 F6C81 000           NIBHEX 000            DATA
200 F6C84 8           CON(1) Comand
201 F6C85 7           NIBHEX 7              Length of DATA
202 F6C86 4414        NIBASC \DATA\
          4514
203             *
204 F6C8E 002         NIBHEX 002
205 F6C91 8           CON(1) Comand
206 F6C92 5           NIBHEX 5              Length of END
207 F6C93 54E4        NIBASC \END\          END
          44
208             *
209 F6C99 006         NIBHEX 006
210 F6C9C 8           CON(1) Comand
211 F6C9D 5           NIBHEX 5              Length of IDY
212 F6C9E 9444        NIBASC \IDY\          IDY
          95
213             *
214             * CoMmanD class...
```

```
215                *
216 F6CA4 F34          NIBHEX F34
217 F6CA7 0            CON(1) Allbit
218 F6CA8 5            NIBHEX 5              Length of UNL
219 F6CA9 55E4         NIBASC \UNL\          UNL
          C4
220                *
221 F6CAF 024          NIBHEX 024
222 F6CB2 4            CON(1) High6
223 F6CB3 B            NIBHEX B              Length of LISTEN
224 F6CB4 C494         NIBASC \LISTEN\       LISTEN
          3545
          54E4
225                *
226 F6CC0 F54          NIBHEX F54
227 F6CC3 0            CON(1) Allbit
228 F6CC4 5            NIBHEX 5              Length of UNT
229 F6CC5 55E4         NIBASC \UNT\          UNT
          45
230                *
231 F6CCB 044          NIBHEX 044
232 F6CCE 4            CON(1) High6
233 F6CCF 7            NIBHEX 7              Length of TALK
234 F6CD0 4514         NIBASC \TALK\         TALK
          C4B4
235                *
236 F6CD8 064          NIBHEX 064
237 F6CDB 4            CON(1) High6
238 F6CDC 5            NIBHEX 5              Length of SAD
239 F6CDD 3514         NIBASC \SAD\          SAD
          44
240                *
241 F6CE3 0A4          NIBHEX 0A4
242 F6CE6 4            CON(1) High6
243 F6CE7 5            NIBHEX 5              Length of DDL
244 F6CE8 4444         NIBASC \DDL\          DDL
          C4
245                *
246 F6CEE 0C4          NIBHEX 0C4
247 F6CF1 4            CON(1) High6
248 F6CF2 5            NIBHEX 5              Length of DDT
249 F6CF3 4444         NIBASC \DDT\          DDT
          45
250                *
251                * CoMmanD class continues below...
252                *
253                * ReaDY class...
254                *
255 F6CF9 005          NIBHEX 005
256 F6CFC 8            CON(1) Comand
257 F6CFD 5            NIBHEX 5              Length of RDY
258 F6CFE 2544         NIBASC \RDY\          RDY
          95
259                *
260                * End of ReaDY class!
```

```
                *
                * At this point, only CoMmanD frames are left...
                *
F6D04 094       NIBHEX 094
F6D07 0         CON(1) Allbit
F6D08 5         NIBHEX 5            Length of IFC
F6D09 9464      NIBASC \IFC\        IFC
      34
                *
F6D0F B94       NIBHEX B94
F6D12 0         CON(1) Allbit
F6D13 5         NIBHEX 5            Length of LPD
F6D14 C405      NIBASC \LPD\        LPD
      44
                *
F6D1A 104       NIBHEX 104
F6D1D 0         CON(1) Allbit
F6D1E 5         NIBHEX 5            Length of GTL
F6D1F 7445      NIBASC \GTL\        GTL
      C4
                *
F6D25 404       NIBHEX 404
F6D28 0         CON(1) Allbit
F6D29 5         NIBHEX 5            Length of SDC
F6D2A 3544      NIBASC \SDC\        SDC
      34
                *
                * End of all defined commands
                *
F6D30 004       NIBHEX 004
F6D33 8         CON(1) Comand
F6D34 5         NIBHEX 5            Length of CMD
F6D35 34D4      NIBASC \CMD\        CMD
      44
                *
                * Following are special case, ASCII search match only!!!!
                * (Will never match on any other search because high bit set)
                *
F6D3B 20F       NIBHEX 20F
F6D3E 0         CON(1) Allbit
F6D3F 5         NIBHEX 5            Length of MLA
F6D40 D4C4      NIBASC \MLA\        MLA (My listen address)
      14
                *
F6D46 40F       NIBHEX 40F
F6D49 0         CON(1) Allbit
F6D4A 5         NIBHEX 5            Length of MTA
F6D4B D445      NIBASC \MTA\        MTA (My talk address)
      14
                *
                * Now all frame types should be complete...put a null entry
                * to end a text search
                *
F6D51 000       NIBHEX 000
F6D54 0         CON(1) Allbit
```

```
309 F6D55 0             NIBHEX 0            Length of last entry (0)
310             *
311             * End of the table!
312             *
313 F6D56               END
```

Allbit	Abs	0	#00000 -	195	217	227	265	270	275	280	295
				300	308						
Comand	Abs	8	#00008 -	192	200	205	210	256	287		
FRAME1	Abs	1010679	#F6BF7 -	71	91						
FRAME2	Abs	1010715	#F6C1B -	94	83						
FRAME3	Abs	1010747	#F6C3B -	117	108						
FRAME4	Abs	1010786	#F6C62 -	137	118						
FRAME8	Abs	1010800	#F6C70 -	148	114	134	138				
FRAME9	Abs	1010807	#F6C77 -	155	78						
=FRAMEE	Abs	1010648	#F6BD8 -	51							
=FRAMET	Abs	1010817	#F6C81 -	190	65						
FRAMSb	Abs	1010655	#F6BDF -	57	56	65					
High6	Abs	4	#00004 -	193	222	232	237	242	247		
High7	Abs	2	#00002 -	194							

Input Parameters

  Source file name is NZ&FRA::MS

  Listing file name is NZ/FRA:TI:ML::-1

  Object file name is NZ%FRA:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *
 2            *       N   N  ZZZZZ   &     L      00000  W   W
 3            *       N   N      Z  & &    L      0   0  W   W
 4            *       NN  N     Z   & &    L      0   0  W   W
 5            *       N N N    Z     &     L      0   0  W W W
 6            *       N  NN   Z     & & &  L      0   0  W W W
 7            *       N   N  Z      &  &   L      0   0  WW WW
 8            *       N   N  ZZZZZ   && &  LLLLL  00000  W   W
 9            *
10            *
11                    TITLE  Low-level USER HP-IL <830927.1414>
12 F6D56              ABS    #F6D56        TI%HP6 address (fixed)
13            ****************************************************************
14            ****************************************************************
15            **
16            ** Name:       FLOAT!,FLOAT+ - Convert a hex value to floating
17            **
18            ** Category:   CONVRT
19            **
20            ** Purpose:
21            **      Converts a hex number into a floating point #
22            **
23            ** Entry:
24            **      FLOAT!: C[W] is the hex value
25            **      FLOAT+: A[W] is the hex value
26            **
27            ** Exit:
28            **      Carry set if value is zero, else clear
29            **      C[W] is the floating number
30            **
31            ** Calls:      HTODX
32            **
33            ** Uses.......
34            **  Exclusive: A[W],     C[W],P
35            **  Inclusive: A[W],B[W],C[W],P
36            **
37            ** Stk lvls:   1 (HTODX)
38            **
39            ** Algorithm:
40            **      FLOAT!:Copy C[W] to A[W]
41            **      FLOAT+:Convert A[W] to decimal                  (HTODX)
42            **             If result is zero, then return, carry set
43            **             Set exponent value (P) to 15 initially
44            **      FLOAT1:Shift result one digit left
45            **             Decrement exponent
46            **             If most significant digit of result = 0 then
47            **               goto FLOAT1
48            **             Shift result right one digit (most sig = 0)
49            **             Put exponent in C[0]
50            **             Return, carry clear (non-zero)
51            **
52            ** History:
53            **
54            **    Date     Programmer             Modification
55            **  --------   ----------    ---------------------------------
```

```
                **  11/19/82      NZ          Added documentation
                **
                ********************************************************************
                ********************************************************************
F6D56 AFA   =FLOAT! A=C     W
F6D59 8E00  =FLOAT+ GOSUBL  =HTODX          Result in B
      00
F6D5F AF9           C=B     W
F6D62 97A   =FLOAT- ?C=0    W               Is initial value 0?
F6D65 00            RTNYES                  Yes...done!
F6D67 2F            P=      15              Initialize exponent to 15
F6D69 BF2   FLOAT1  CSL     W               Shift result left one digit
F6D6C 0D            P=P-1                   Decrement exponent
F6D6E 94A           ?C=0    S               Is most significant digit zero?
F6D71 8F            GOYES   FLOAT1          Yes...loop back for more
F6D73 BF6           CSR     W               No...undo last shift (C[S]=0)
F6D76 AB2           C=0     X               Clear exponent field
F6D79 80F0          CPEX    0               Set C[0] to exponent value
F6D7D 20            P=      0               (Unnecessary instruction)
F6D7F 03            RTNCC                   Return, carry clear (non-zero)
                ********************************************************************
                ********************************************************************
                **
                ** Name:       POP1N - Pop one numeric value from MTHSTK
                **
                ** Category:   GETUTL
                **
                ** Purpose:
                **      (Same as mainframe POP1N)
                **
                ** Entry:
                **      D1 points to top of stack
                **
                ** Exit:
                **      DECIMAL MODE!!!
                **      P=0
                **      If not numeric, jumps to ERRORX
                **      A[W] is real part, R0 is imaginary (if complex)
                **      Carry clear if real, carry set if complex
                **
                ** Calls:      None
                **
                ** Uses.......
                **  Inclusive: A[W],B[0],R0,D1,P
                **
                ** Stk lvls:   0
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------    --------------------------------
                **  11/19/82      NZ          Added documentation
                **
                ********************************************************************
                ********************************************************************
```

```
F6D81 05    =POP1N  SETDEC
F6D83 20            P=     0
F6D85 A81           B=0    P
F6D88 A0D           B=B-1  P             Set B[0]=9
F6D8B 1537          A=DAT1 W             Read the item
F6D8F 980           ?A>B   P
F6D92 40            GOYES  POP1N#        Check if complex or otherwise
F6D94 03            RTNCC
            *-
            *-
F6D96 04    POP1N#  SETHEX
F6D98 B04           A=A+1  P
F6D9B B04           A=A+1  P
F6D9E 96C           ?A#0   B             Check if complex (0E)
F6DA1 71            GOYES  POP1NE        Error...type conflict
F6DA3 171           D1=D1+ 2
F6DA6 1537          A=DAT1 W             Read in imaginary part
F6DAA 17F           D1=D1+ 16
F6DAD 100           R0=A                 Save im part in R0
F6DB0 1537          A=DAT1 W             Read in real part
F6DB4 05            SETDEC
F6DB6 02            RTNSC                Return with carry SET
            *-
            *-
F6DB8 20    POP1NE  P=     =eNNUMR       Not numeric
F6DBA 8C00  Errorx  GOLONG =ERRORX
      00
            ****************************************************************
            ****************************************************************
            **
            ** Name:       RESET - Reset the Diamond processor
            **
            ** Category:   STEXEC
            **
            ** Purpose:
            **      Reset an HPIL mailbox (Diamond), set up default parms
            **
            ** Entry:
            **      None
            **
            ** Exit:
            **      Through NXTSTM
            **
            ** Calls:      GTYPRM,GETLOP,FNDMB-,GETERR,CHKST+
            **
            ** Uses.......
            **  Exclusive: A,  C,                        P
            **  Inclusive: A,B,C,D,R0,R1,R2,R3,R4,D0,D1,P,ST[11:0],FUNCxx
            **
            ** Stk lvls:   6 (GTYPRM)
            **
            ** History:
            **
            **    Date     Programmer              Modification
            **  --------   ----------    ---------------------------------
```

```
164                ** 09/26/83      NZ          Updated documentation
165                ** 06/24/83      NZ          Changed to wake up diamond and
166                **                           REPORT any errors it found
167                ** 11/19/82      NZ          Added documentation
168                **
169                ********************************************************************
170                ********************************************************************
171 F6DC0 00000          REL(5) =RESETd
          0
172 F6DC5 00000          REL(5) =RESETp
          0
173 F6DCA       =RESET
174 F6DCA AC2            C=0    S              Clear C[S]
175 F6DCD 14A            A=DAT0 B              Check if a loop # given
176 F6DD0 3100           LC(2)  =tCOMMA
177 F6DD4 962            ?A=C   B
178 F6DD7 11             GOYES  RESET0         Not loop expression...skip it
179 F6DD9 8E00           GOSUBL =GTYPRM        Get (type) loop # from RAM
          00
180 F6DDF 453            GOC    Resete         If out of range, error
181 F6DE2 8E00           GOSUBL =GETLOP        Get loop # into C[S]
          00
182 F6DE8       RESET0
183 F6DE8 8E00           GOSUBL =FNDMB-        Clear DISPLAY, etc; FNDMBX
          00
184 F6DEE 462            GOC    Resete         If not found, error!
185 F6DF1 AF2            C=0    W
186 F6DF4 27             P=     7
187 F6DF6 308            LCHEX  8              Reset Diamond
188 F6DF9 15C8           DAT0=C 9              Clear the NRD bit, too!
189 F6DFD 7000           GOSUB  =GETERR        Check if any errors
190 F6E01 431            GOC    Resete         Errors!
191 F6E04 8E00           GOSUBL =CHKST+        Set up parameters for Diamond
          00
192 F6E0A 4A0            GOC    Resete         Error
193 F6E0D 8C00 Nxtstm    GOLONG =nXTSTM        Next BASIC statement
          00
194                *-
195                *-
196 F6E13 20   Resetr    P=     =eRANGE        (Unnecessary instruction)
197 F6E15 64AF Resete    GOTO   Errorx
198 F6E19                END
```

```
 CHKST+  Ext                  -   191
 ERRORX  Ext                  -   135
 Errorx  Abs 1011130 #F6DBA   -   135   197
=FLOAT!  Abs 1011030 #F6D56   -    60
=FLOAT+  Abs 1011033 #F6D59   -    61
=FLOAT-  Abs 1011042 #F6D62   -    63
 FLOAT1  Abs 1011049 #F6D69   -    66    69
 FNDMB-  Ext                  -   183
 GETERR  Ext                  -   189
 GETLOP  Ext                  -   181
 GTYPRM  Ext                  -   179
 HTODX   Ext                  -    61
 Nxtstm  Abs 1011213 #F6E0D   -   193
=POP1N   Abs 1011073 #F6D81   -   110
 POP1N#  Abs 1011094 #F6D96   -   120   116
 POP1NE  Abs 1011128 #F6DB8   -   134   124
=RESET   Abs 1011146 #F6DCA   -   173
 RESET0  Abs 1011176 #F6DE8   -   182   178
 RESETd  Ext                  -   171
 RESETp  Ext                  -   172
 Resete  Abs 1011221 #F6E15   -   197   180   184   190   192
 Resetr  Abs 1011219 #F6E13   -   196
 eNNUMR  Ext                  -   134
 eRANGE  Ext                  -   196
 nXTSTM  Ext                  -   193
 tCOMMA  Ext                  -   176
```

Input Parameters

  Source file name is NZ&LOW::MS

  Listing file name is NZ/LOW:TI:ML::-1

  Object file name is NZ%LOW:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
                *
                *
                *       N   N  ZZZZZ   &     FFFFF  X   X   QQQ
                *       N   N      Z  & &    F      X   X  Q   Q
                *       NN  N     Z   & &    F       X X   Q   Q
                *       N N N    Z     &     FFFF     X    Q   Q
                *       N  NN   Z     & & &  F       X X   Q Q Q
                *       N   N  Z      &  &   F      X   X  Q  Q
                *       N   N  ZZZZZ   && &  F      X   X   QQ Q
                *
                *
                        TITLE  File Execution <840113.1351>
F6E19           ABS    #F6E19        TI%HP6 address (fixed)
                ****************************************************************
                ****************************************************************
                **
                ** Name:       GETDID - Get device ID (specifier)
                ** Name:       GETDIX - Get device ID (String expr on stack)
                **
                ** Category:   FILUTL
                **
                ** Purpose:
                **      GETDID fetches a device ID, given D0 pointing to the
                **      ID in program memory
                **
                ** Entry:
                **      D0 points to the ID in program memory
                **
                ** Exit:
                **      Carry clear: Address/type in D[X], device type/ID in B
                **        If D[X]=0, then device id = "" OR *
                **        P=0
                **        FUNCD0 contains the D0 value after evaluating ID
                **      Carry set: error, P=error number
                **
                ** Calls:      GETSTR,PROCLT,NXTCHR,BAKCHR,PROCST,TSAVD0,START
                **
                ** Uses.......
                **  Inclusive: A-D,R0-R4,D0,D1,P,STMTD1[3:0],STMTR1,FUNCxx,
                **             ST[11:0],all RAM that EXPEXC is permitted to use
                **
                ** Stk lvls:   GETDID: 6 (GETSTR)
                ** Stk lvls:   GETDIX: 4 (PROCST)
                **
                ** History:
                **
                **    Date     Programmer          Modification
                **  --------   ----------   --------------------------------
                **  05/02/83      NZ        Added flag for colon/semicolon
                **                          required to GETDIX, added GETDI+
                **  03/18/83      NZ        Changed GETDIX to use NXTCHR,
                **                          removed SaveD0 code
                **  03/17/83      NZ        Changed register usage (+STMTD1,
                **                          remove STMTR0)
                **  03/15/83      NZ        Returned exit conditions to those
```

```
                **                              originally given (D[A])
                **  03/01/83      NZ            Changed GETDIX to use PROCLT
                **  11/04/82      NZ            Added documentation
                **
                ****************************************************************
                ****************************************************************
                TermRq  EQU   0               Status bit for terminator required
                *
F6E19 8E00 =GETDID GOSUBL =GETSTR             Get string/literal-sets =ST(sSTK)
      00
F6E1F 400       RTNC                          If carry, ERROR
F6E22 870       ?ST=1 =sSTK
F6E25 D0        GOYES GETDI1                  String expression
                *
                * Literal expression in memory
                *
F6E27 7834      GOSUB PROCLT                  Process literal
F6E2B D7        D=C   A                       Put device type into D[A]
F6E2D 555       GONC  GETDI5                  If no carry, finish it up
F6E30 02        RTNSC                         If carry, error
                *-
                *-
F6E32           GETDI1
                *
                * This is a string expression
                * (Start of string in D1, D[A] @ end of string)
                *
F6E32 8A8       ?A=0  A
F6E35 C4        GOYES GETDI4                  Null string
F6E37 840  =GETDIX ST=0  TermRq               Terminator (colon/semic) optional
F6E3A 7B56 GETDI+ GOSUB Nxtchr                Read the first char
F6E3E 4E3       GOC   GETDI3                  End of string...error
                *
                * Is it a ":"?
                *
F6E41 31A3      LCASC \:\
F6E45 962       ?A=C  B                       Is it a colon?
F6E48 41        GOYES GETDIO                  Yes...Nxtchr was OK
F6E4A 31E2      LCASC \.\                     No...check volume label
F6E4E 962       ?A=C  B                       Is it a volume label?
F6E51 31        GOYES GETDI2                  Yes...process volume label
F6E53 870       ?ST=1 TermRq                  Was a terminator required?
F6E56 72        GOYES GETDI3                  Yes...bad device spec
F6E58 7276      GOSUB Bakchr                  No...back it up
F6E5C 70F0 GETDIO GOSUB PROCST                Process string entry point
F6E60 6700      GOTO  GETDCK
                *-
                *-
F6E64 7B91 GETDI2 GOSUB PRSTv1                Yes...process volume label
                *
                * Fall into GETDCK
                *
F6E68 400  GETDCK RTNC                        If carry, error
F6E6B 7A26      GOSUB Nxtchr                  Check to be sure no more data
F6E6F D7        D=C   A                       Put address in D[A]
```

```
110 F6E71 411          GOC    GETDI5        If carry, end of string
111 F6E74 3102         LCASC  \ \
112 F6E78 962          ?A=C   B             Is the next char a blank?
113 F6E7B 80           GOYES  GETDI5        Yes...accept it
114 F6E7D 20   GETDI3  P=     =eDSPEC       Illegal device id
115 F6E7F 02           RTNSC
116            *-
117            *-
118 F6E81 D3   GETDI4  D=0    A
119 F6E83 8E00 GETDI5  GOSUBL =TSAVDO       Save DO in FUNCDO
           00
120 F6E89 850          ST=1   =sDevOK       If here, device is OK
121 F6E8C 8AB          ?D=0   A
122 F6E8F EE           GOYES  GETDI3        If D=0, then "", "*", or *
123 F6E91 8E00         GOSUBL =START        Find out the tape address
           00
124 F6E97 400          RTNC                 Error
125 F6E9A 8C00         GOLONG =SETUP        Arrange the info from START
           00
126            ****************************************************************
127            ****************************************************************
128            **
129            ** Name:       GETPIL - Evaluate an HPIL file specifier
130            ** Name:       GETPI+ - Get an HPIL file specifier from stack
131            **
132            ** Category:   FILUTL
133            **
134            ** Purpose:
135            **      This routine extracts the file name and the device
136            **      and returns with the device type/device ID in B[W],
137            **      address/type in D[X]
138            **
139            ** Entry:
140            **      DO points to the file specifier in program memory
141            **
142            ** Exit:
143            **      ST(sDevOK) set if device spec was ok, else clear
144            **      Carry clear:
145            **        Filename in R0, R4[15:12]
146            **        Device type in B[X]/B[W], address in D[X]
147            **        If address = X00, then this is a * or a ""
148            **        AVMEME collapsed back to starting point
149            **      Carry set:
150            **        Error (P,C[0] are error code)
151            **
152            ** Calls:      GETSTR,FXQPIL,NXTCHR,PROCLT,PROCST,ASRC4,D1=AVS,
153            **             D1@AVE,CSRC12,GETDI5,ASLC12
154            **
155            ** Uses.......
156            **  Inclusive: A-D,R0-R4,D0,D1,P,STMTD1[3:0],STMTR1,ST[11:0],
157            **             FUNCxx,all RAM that EXPEXC is permitted to use
158            **
159            ** Stk lvls:   6 (GETSTR)
160            **
161            ** History:
```

```
                **
                **    Date     Programmer          Modification
                **  --------   ----------   --------------------------------
                **  05/01/83      NZ        Changed GOSUB GETDIX to GETDI+,
                **                          added ST=1 TermRq
                **  03/17/83      NZ        Changed STMT usage (+D1, -R0)
                **  03/01/83      NZ        Changed GOYES GETDI2 to GETDIX,
                **                          added call to GETDCK
                **  11/04/82      NZ        Added documentation
                **
                ****************************************************************
                ****************************************************************
F6EA0 8E00 =GETPIL GOSUBL =GETSTR       Get string/literal
      00
F6EA6 400          RTNC                 Error
F6EA9 7735 =GETPI+ GOSUB  FXQPIL
                *
                * FXQPIL returns with filename (blank-filled) in R0,A[3:0]
                * (If carry set, A,R0 are zeroed out)
                *
                * ST(sSTK) is set if reading from the stack, clear if prog mem
                *
                * Now the filename is in A and R0
                *
                * Move the last two characters to A[15:12], than to R4
                *
F6EAD 8E00         GOSUBL =ASRC4
      00
F6EB3 11C          C=R4
F6EB6 2B           P=     11
F6EB8 A9E          ACEX   WP
F6EBB 104          R4=A
                *
                * If sSTK is 1, then reading from the stack...process stack
                *
F6EBE 20           P=     0
F6EC0 840          ST=0   =sDevOK       Device spec NOT ok until shown so
F6EC3 860          ?ST=0  =sSTK         Stack?
F6EC6 90           GOYES  GETPI1        No...continue...
F6EC8 850          ST=1   TermRq        Terminator (:;) required
F6ECB 6E6F         GOTO   GETDI+        Read it from the stack
                *-
                *-
                *
                * Need to save filename on stack to protect from PROCLT
                *
F6ECF 8E00 GETPI1  GOSUBL =D1=AVS       Set D1 = AVMEMS
      00
F6ED5 143          A=DAT1 A             A[A] is @ AVMEMS
F6ED8 D2           C=0    A
F6EDA 3141         LC(2)  20            20 nibs for the filename
F6EDE D5           B=C    A             Save in B[A] for now
F6EE0 8E00         GOSUBL =D1@AVE       Set D1 @ AVMEME, C[A] = AVMEME
      00
F6EE6 137          CD1EX                D1=AVMEME,C[A] @ AVMEME
```

```
 213 F6EE9 E9          C=C-B  A            C[A] is proposed new AVMEME
 214 F6EEB 8B6         ?A>C   A            Enough memory?
 215 F6EEE A5          GOYES  GETPIn       No...insufficient memory
 216 F6EF0 145         DAT1=C A            Yes...write out new AVMEME
 217 F6EF3 135         D1=C                Set D1 @ AVMEME
 218 F6EF6 118         C=RO
 219 F6EF9 1557        DAT1=C W            Write out first 8 chars of name
 220 F6EFD 17F         D1=D1+ 16
 221 F6F00 11C         C=R4
 222 F6F03 8E00        GOSUBL =CSRC12
           00
 223 F6F09 15D3        DAT1=C 4            Write out last 2 chars of name
 224               *
 225               * Done saving name on stack
 226               *
 227 F6F0D 7253        GOSUB  PROCLT       Process literal
 228 F6F11 400         RTNC                Error (leaves info on MTHSTK)
 229 F6F14 D7          D=C    A            Put device type into D[A]
 230 F6F16 796F        GOSUB  GETDI5       Check it and set it up
 231 F6F1A 400         RTNC
 232               *
 233               * Now restore the filename from stack
 234               *
 235 F6F1D 06          RSTK=C              Save C[A] on RSTK
 236 F6F1F 8E00        GOSUBL =D1@AVE      (C[A] = AVMEME)
           00
 237 F6F25 1537        A=DAT1 W
 238 F6F29 17F         D1=D1+ 16
 239 F6F2C 100         RO=A                Restore first 8 chars to RO
 240 F6F2F 143         A=DAT1 A
 241 F6F32 8E00        GOSUBL =RSLC12      Put last 2 chars in R4[15:12]
           00
 242 F6F38 104         R4=A
 243 F6F3B 173         D1=D1+ 4
 244 F6F3E 137         CD1EX               Set D1 = AVMEME
 245 F6F41 145         DAT1=C A            Write out new AVMEME (pop 20)
 246 F6F44 07          C=RSTK              Restore C[A]
 247               *
 248               * Done restoring levels now
 249               *
 250 F6F46 03          RTNCC
 251               *-
 252               *-
 253 F6F48 20    GETPIn P=     =eNORAM
 254 F6F4A 02          RTNSC               Error...no memory
 255               ********************************************************************
 256               ********************************************************************
 257               **
 258               ** Name:       PROCST - Process string device specifier
 259               **
 260               ** Category:   FILUTL
 261               **
 262               ** Purpose:
 263               **      Process a device specifier from a string expression
 264               **
```

```
                ** Entry:
                **      ST(sSTK)=1
                **      R0[W], R4[15:12] are filename
                **      D1 points to next item of string
                **      D[A] is the end of the string
                **      HEXMODE
                **
                ** Exit:
                **      Carry set if error (P,C[0] are error number)
                **      Carry clear:
                **              P=0
                **              Device type/device id in B[X]/B[W]
                **              IF device type="*", *, or "" THEN C[X]=0
                **              ELSEIF address, THEN C[X] is address+loop*1024
                **              ELSEIF LOOP, THEN C[X] is "9F"+loop*4096
                **              ELSEIF NULL, THEN C[B] is "7F"
                **              ELSEIF volume label THEN C[X] is "5F"+loop*4096
                **              ELSEIF device type THEN C[X] is "3F"+loop*4096
                **              ELSEIF device id THEN C[X] is "1F"+loop*4096
                **
                ** Calls:       NXTCHR,BAKCHR,UCRANG,GETDVW,PROCDW,GTYPST,GADRST
                **
                ** Uses.......
                **  Exclusive: A[W],B[W],C[W],R1,R2,   P
                **  Inclusive: A[W],B[W],C[W],R1,R2,D1,P
                **
                ** Stk lvls:   3 (GETDVW)
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------    ---------------------------------
                **  11/04/82      NZ          Added documentation
                **
                ********************************************************************
                ********************************************************************
F6F4C 20        PRSTed  P=      =eDSPEC       Error...device spec
F6F4E 02                RTNSC
                *-
                *-
F6F50           =PROCST                       Process string device spec
F6F50 7545              GOSUB   Nxtchr
F6F54 47F               GOC     PRSTed        No device spec
F6F57 7C95              GOSUB   Ucrang        Convert upper case, check [A-Z]
F6F5B 497               GOC     PRST30        Not in [A-Z]
                *
                * Character IS in [A-Z]...continue
                *
                * Assign word, reserved word, or device id
                *
F6F5E 7C65              GOSUB   Bakchr        Back past the character
F6F62 7262              GOSUB   GETDVW        Get device word
F6F66 45E               GOC     PRSTed        Bad device word (Error)
F6F69 78A2              GOSUB   PROCDW        Process device word
F6F6D 460               GOC     PRST10        If carry, takes a seq number
```

```
320 F6F70 6F90          GOTO   PRST90        If no carry, does NOT take seq #
321             *-
322             *-
323             *
324             * Now process sequence #
325             *
326 F6F74 109   PRST10  R1=C                  Save type in R1
327 F6F77 AF9           C=B    W
328 F6F7A 10A           R2=C                  Save type/ID in R2
329             *
330             * Get sequence #
331             *
332 F6F7D 7815          GOSUB  Nxtchr
333 F6F81 4A3           GOC    PRST25         No sequence number...continue
334 F6F84 20            P=     0
335 F6F86 3182          LCASC  \(\
336 F6F8A 966           ?A#C   B
337 F6F8D B2            GOYES  PRST20         No sequence #...back up, continue
338             *
339             * This has a sequence number...get it
340             *
341 F6F8F 75F0          GOSUB  GTYPST         Get type
342 F6F93 400           RTNC                  Error
343 F6F96 7FF4          GOSUB  Nxtchr
344 F6F9A 41B           GOC    PRSTed         No closing ")"...error
345             *
346             * Check for closing parenthesis
347             *
348 F6F9D 3192          LCASC  \)\
349 F6FA1 966           ?A#C   B
350 F6FA4 8A            GOYES  PRSTed         Error...no closing ")"
351             *
352             * Closed properly...check its range
353             *
354             * First convert to zero-based count
355             *
356 F6FA6 D9            C=B    A              Copy 2 digits to C[A]
357 F6FA8 CE            C=C-1  A              convert to zero-based
358 F6FAA 490           GOC    PRSTeR         Range error
359 F6FAD 21            P=     1              Check that C[1]=0
360 F6FAF 90A           ?C=0   P
361 F6FB2 C0            GOYES  PRST27         Go always...continue
362             *
363 F6FB4 20    PRSTeR  P=     =eRANGE
364 F6FB6 02            RTNSC
365             *-
366             *-
367 F6FB8 7215  PRST20  GOSUB  Bakchr         Back up 1 character
368 F6FBC D2    PRST25  C=0    A
369             *
370             * Now C[B] is sequence #
371             *
372 F6FBE 112   PRST27  A=R2                  Recall type/ID
373 F6FC1 AF8           B=A    W
374 F6FC4 111           A=R1
```

```
375 F6FC7 F2           CSL   A
376 F6FC9 F2           CSL   A            Sequence # is in C[XS] now
377 F6FCB AE6          C=A   B            Type/ID in C[B] now
378 F6FCE 21           P=    1
379 F6FD0 0D           P=P-1              Clear the carry...
380 F6FD2 5D3          GONC  PRST90       Done
381             *-
382             *-
383 F6FD5 31A2 PRST30  LCASC \*\
384 F6FD9 966          ?A#C  B            Is this a "*"?
385 F6FDC 70           GOYES PRST40       No...continue
386             *
387             * Device spec is "*"
388             *
389 F6FDE D2           C=0   A            Yes...continue with C[A]=0
390 F6FE0 5F2          GONC  PRST90       Go always...carry clear
391             *-
392             *-
393 F6FE3 3152 PRST40  LCASC \%\
394 F6FE7 966          ?A#C  B            Is this a device type?
395 F6FEA D0           GOYES PRST50       No...must be address
396             *
397             * Device type
398             *
399 F6FEC 7890         GOSUB GTYPST       Get type from stack
400 F6FF0 4F1          GOC   PRST90       If carry, error
401 F6FF3 608F         GOTO  PRST10       Process sequence #
402             *-
403             *-
404 F6FF7      PRST50
405             *
406             * Address...back up to first character
407             *
408 F6FF7 73D4         GOSUB Bakchr       Back up 1 character
409 F6FFB 7AF0         GOSUB GADRST       Get address from stack
410 F6FFF 6010         GOTO  PRST90       Carry indicates status
411             *-
412             *-
413             *
414             * Process string volume spec
415             *
416 F7003 71C1 PRSTv1  GOSUB GETDVW       Get volume word (get device word)
417 F7007 400          RTNC               Carry if error
418 F700A D2           C=0   A            Clear high nibbles of C[A]
419 F700C 3100         LC(2) =VolLbl      Volume label identifier
420             *
421             * Check if a loop spec here...
422             *
423 F7010 400  PRST90  RTNC               If carry, error in C[0], P
424 F7013 109          R1=C               Save address/type in R1
425 F7016 AF9          C=B   W
426 F7019 10A          R2=C               Save device in R2
427 F701C 7974         GOSUB Nxtchr
428 F7020 4C5          GOC   PROCex       Exit...done
429 F7023 31A3         LCASC \:\
```

```
430 F7027 966          ?A#C   B
431 F702A F4           GOYES  PROCeX        Not a loop spec...exit
432             *
433             * Have a loop spec
434             *
435             * Process string loop spec
436             *
437 F702C 7850         GOSUB  GTYPST        Get type from stack
438 F7030 400          RTNC                 Error (Bad loop #)
439             *
440             * Now loop # is in B[A]
441             *
442 F7033 3130         LC(2)  3             ...maximum value is 3
443 F7037 9E1          ?B>C   B
444 F703A 90           GOYES  PRSTer        Out of range
445 F703C D9           C=B    A             Copy back to C[B]
446 F703E CE           C=C-1  A             Convert to zero-based count
447 F7040 560          GONC   PRSTEX        If no carry, all OK
448             *
449             * If carry, out of range
450             *
451 F7043 20   PRSTer  P=     =eRANGE
452 F7045 02          RTNSC
453             *-
454             *-
455             *
456             * Now integrate loop spec with device spec
457             *
458 F7047 112  PRSTEX  A=R2
459 F704A AF8         B=A    W             Restore device ID
460 F704D 111         A=R1                 Recall type
461 F7050 816         CSRC                 Save loop # in C[S]
462 F7053 310E        LCHEX  E0            Check if not address
463 F7057 0E6A        C=A!C  B
464 F705B B66         C=C+1  B             If carry, not address
465 F705E 812         CSLC
466 F7061 F2          CSL    A
467 F7063 F2          CSL    A
468 F7065 4C0         GOC    PROCna        Not address
469             *
470             * Address...multiply times 4
471             *
472 F7068 C6          C=C+C  A
473 F706A C6          C=C+C  A
474 F706C 0E3A        C=C!A  X             Now C[X] is loop #, address
475 F7070 03          RTNCC
476             *-
477             *-
478 F7072 F2   PROCna  CSL    A
479 F7074 AB6         C=A    X             Loop # in C[3], device in C[X]
480 F7077 03          RTNCC
481             *-
482             *-
483 F7079 7154 PROCeX  GOSUB  Bakchr        Back up last character fetch
484 F707D 11A  PROCex  C=R2                 Recall device
```

```
485 F7080 AF5          B=C    W
486 F7083 119          C=R1                  Recall type/address
487 F7086 03           RTNCC                 Done
488             ****************************************************************
489             ****************************************************************
490             **
491             ** Name:       GTYPST - Get type from stack
492             **
493             ** Category:   FILUTL
494             **
495             ** Purpose:
496             **      Given a pointer to the start of the type, return the
497             **      numeric value of the type
498             **
499             ** Entry:
500             **      D1 @ first digit of type
501             **      D[A] @ end of specifier
502             **
503             ** Exit:
504             **      Carry clear:
505             **        Type in B[X], D1 @ first unused item
506             **        C[X]=(=DevTyp)
507             **        P=0
508             **      Carry set:
509             **        error (P, C[0] are error code)
510             **
511             ** Calls:      NXTCHR,BAKCHR,DTOH,RANGEN
512             **
513             ** Uses.......
514             **  Exclusive: A[W],B[W],C[W],   P
515             **  Inclusive: A[W],B[W],C[W],D1,P
516             **
517             ** Stk lvls:   1 (NXTCHR)(BAKCHR)(DTOH)(RANGEN)
518             **
519             ** History:
520             **
521             **    Date     Programmer            Modification
522             **  --------   ----------   ---------------------------------
523             **  11/04/82      NZ        Added documentation
524             **
525             ****************************************************************
526             ****************************************************************
527 F7088 AF1  =GTYPST B=0    W             Clear B[W] (where total is built)
528 F708B 7A04 GTYPS1  GOSUB  Nxtchr        Get next character
529 F708F 405          GOC    GTYPS5        End of string
530 F7092 7234         GOSUB  Rangen        Check if in [0-9]
531 F7096 401          GOC    GTYPS3        No...done?
532             *
533             * New digit...add it in
534             *
535 F7099 F1           BSL    A
536 F709B A88          B=A    P             Append new digit here
537 F709E 959          ?B=0   M             If non-zero, too big
538 F70A1 AE           GOYES  GTYPS1        Zero...continue
539             *
```

```
                        * Out of range
                        *
F70A3 20    GTYPS2  P=     =eRANGE
F70A5 02            RTNSC
                        *-
                        *-
F70A7 31E2  GTYPS3  LCASC  \.\
F70AB 966           ?A#C   B
F70AE E2            GOYES  GTYPS4        Not a period...exit
                        *
                        * Got a period...continue
                        *
F70B0 75E3          GOSUB  Nxtchr
F70B4 4B2           GOC    GTYPS5        End of string
F70B7 7D04          GOSUB  Rangen        Check if in [0-9]
F70BB 402           GOC    GTYPS4        No...exit
F70BE 05            SETDEC
F70C0 A04           A=A+A  P             Check if round UP
F70C3 550           GONC   GTYPS.        No...exit
F70C6 B35           B=B+1  X
F70C9 04    GTYPS.  SETHEX               (jump to here has carry CLEAR!)
F70CB 47D           GOC    GTYPS2        Error...overflow
                        *
                        * Loop to skip trailing digits
                        *
F70CE 77C3  GTYPSd  GOSUB  Nxtchr
F70D2 4D0           GOC    GTYPS5        End of string
F70D5 7FE3          GOSUB  Rangen        Check if digit
F70D9 54F           GONC   GTYPSd        Yes...continue
                        *
F70DC 7EE3  GTYPS4  GOSUB  Bakchr        Back up past the last character
F70E0 AF4   GTYPS5  A=B    W             Convert it to HEX now
F70E3 8E00          GOSUBL =DTOH
      00
                        *
                        * Now the type is in C (in HEX)
                        *
F70E9 D1            B=0    A             Check if in [0,255]
F70EB AED           BCEX   B             (B[A] is value if C=0)
F70EE 8AE           ?C#0   A
F70F1 2B            GOYES  GTYPS2        Out of range
                        *
                        * C[A] is zero to get here
                        *
F70F3 3100          LC(2)  =DevTyp       Device type
F70F7 03            RTNCC
                        ****************************************************************
                        ****************************************************************
                        **
                        ** Name:       GADRST - Get address from stack
                        **
                        ** Category:   FILUTL
                        **
                        ** Purpose:
                        **      Similar to GTYPST, except that the first 2 digits
```

```
                **      after the decimal point, if any, are used as the
                **      secondary address
                **
                ** Entry:
                **      D1 @ first character
                **      D[A] @ end of spec
                **
                ** Exit:
                **      Carry clear:
                **        C[X] is address
                **        D1 @ first unused character
                **        Skips trailing digits
                **        P=0
                **      Carry set:
                **        P, C[0] are error code
                **
                ** Calls:      NXTCHR,BAKCHR,RANGEN,DTOH,CSRC2
                **
                ** Uses.......
                **  Exclusive: A,B,C,   P
                **  Inclusive: A,B,C,D1,P
                **
                ** Stk lvls:   1 (NXTCHR)(BAKCHR)(RANGEN)(DTOH)(CSRC2)
                **
                ** Algorithm:
                **      Read a number from the stack until non-digit OR full;
                **      Check if "."...if not, return
                **      Get another number from the stack (2 digits)
                **      Combine the two numbers as one address, return
                **
                ** History:
                **
                **    Date      Programmer             Modification
                **  --------    ----------    --------------------------------
                **  12/21/83      NZ          Changed order of BSL A, ?B=0 XS
                **                            test at GADRS4 to fix a bug which
                **                            got into an infinite loop.  If the
                **                            device spec contained a ".000x",
                **                            where "x" is not a digit, the
                **                            code at GADRS4 would end up with
                **                            B[X]=0, which caused an infinite
                **                            assembly code loop.
                **  11/04/82      NZ          Added documentation
                **
                ********************************************************************
                ********************************************************************
F70F9 AF1       =GADRST B=0     W             Clear B[W] to start
F70FC 7993      GADRS1  GOSUB   Nxtchr         Get first item
F7100 442               GOC     GADRS.        End of string...continue process
F7103 71C3              GOSUB   Rangen        Check if in [0-9]
F7107 401               GOC     GADRS2        No...check further
F710A F1                BSL     A
F710C A88               B=A     P             Copy this digit in
F710F 929               ?B=0    XS            Overflow?
F7112 AE                GOYES   GADRS1        No...continue
```

```
649 F7114 20   GADRSo  P=     =eRANGE
650 F7116 02           RTNSC
651            *-
652            *-
653 F7118      GADRS2
654            *
655            * Got a non-digit...if not a decimal point, done
656            *
657 F7118 31E2         LCASC  \.\
658 F711C 962          ?A=C   B
659 F711F 60           GOYES  GADRS.        "."...continue
660            *
661 F7121 79A3         GOSUB  Bakchr        Back up for next step
662            *
663            * Decimal point...get secondary address
664            *
665 F7125 AF4  GADRS.  A=B    W
666 F7128 8E00         GOSUBL =DTOH         Convert primary address to hex
          00
667            *
668            * Hex value in C[B] now
669            *
670 F712E 8E00         GOSUBL =CSRC2        Use C[15:14] as temp storage
          00
671            *
672            * Primary address in C[15:14] now
673            *
674 F7134 AF5          B=C    W             Copy to B[15:14]
675 F7137 D1           B=0    A             Clear B[0]
676 F7139 E5           B=B+1  A             Set B[0]=1 (Flag for 2 digits)
677 F713B 7A53 GADRS3  GOSUB  Nxtchr        Get next character
678 F713F 434          GOC    GADRS4        End...manipulate it
679 F7142 7283         GOSUB  Rangen        Check if in [0-9]
680 F7146 483          GOC    GADRSb        No...back up, manipulate it
681 F7149 F1           BSL    A
682 F714B A88          B=A    P             Copy to B
683 F714E 929          ?B=0   XS            Done yet?
684 F7151 AE           GOYES  GADRS3        No...continue
685            *
686            * Reached here by reading 2 digits after decimal point
687            *
688 F7153 7243         GOSUB  Nxtchr        Get next digit for rounding
689 F7157 493          GOC    GADRS6        No next digit...continue
690 F715A 7A63         GOSUB  Rangen        Check if in [0-9]
691 F715E 4E2          GOC    GADRS5        Not a digit...back it up
692 F7161 05           SETDEC
693 F7163 A04          A=A+A  P             Check if rounding needed
694 F7166 550          GONC   GADRSs        Skip other digits
695 F7169 B65          B=B+1  B             Round UP
696 F716C 04   GADRSs  SETHEX
697 F716E 45A          GOC    GADRSo        Out of range (If B=B+1 carry)
698 F7171 7423         GOSUB  Nxtchr        Read next character
699 F7175 4B1          GOC    GADRS6        (End of string)
700 F7178 7C43         GOSUB  Rangen        Check if a digit
701 F717C 5FE          GONC   GADRSs        Yes...skip the next one
```

```
                        *                               No...fall through to GADRSb
F717F 7B43  GADRSb  GOSUB  Bakchr        Back up the last NXTCHR
F7183       GADRS4
                        *
                        * Reached here before two digits
                        *
                        * B[X] cannot be zero to get here...at least one digit of B[X]
                        * must be 1 (from flag set before GADRS3)
                        *
F7183 92D           ?B#0   XS            Done yet?
F7186 B0            GOYES  GADRS6        Yes
F7188 F1            BSL    A             Shift in a zero
F718A 58F           GONC   GADRS4        Go always
                        *-
                        *-
F718D 7D33  GADRS5  GOSUB  Bakchr        Back up the last NXTCHR
F7191       GADRS6
                        *
                        * Now B[B] is secondary address in decimal...convert to hex
                        *
F7191 D0            A=0    A
F7193 AE4           A=B    B
F7196 8E00          GOSUBL =DTOH
          00
F719C AE5           B=C    B
                        *
                        * Now B[B] is secondary address in hex, B[15:14] is primary
                        *
F719F 31F1          LC(2)  31
F71A3 9E1           ?B>C   B             >31?
F71A6 6C            GOYES  GADRSs        Too big for secondary!(Jump jump)
F71A8 811           BSLC
F71AB 811           BSLC                 Now B[B] is primary address
F71AE 9E9           ?B>=C  B             >30?
F71B1 BB            GOYES  GADRSs        Too big for primary! (Jump jump)
F71B3 969           ?B=0   B
F71B6 6B            GOYES  GADRSs        Zero is NOT a legal primary addr
                        *
                        * B[B] is primary, B[3:2] is secondary
                        *
F71B8 D2            C=0    A             Clear C[XS]
F71BA AED           CBEX   B             Copy primary to C[B], zero B[B]
F71BD F5            BSR    A             Secondary in B[2:1]
F71BF A35           B=B+B  X             Secondary*2 in B[2:1]
F71C2 0E3D          C=C!B  X             Primary, secondary in C[X]
                        *
                        * Now address is in C[A]
                        *
F71C6 03            RTNCC
                        ******************************************************************
                        ******************************************************************
                        **
                        ** Name:       GETDVW - Get device word
                        **
                        ** Category:   FILUTL
```

```
                **
                ** Purpose:
                **      Get a device word, given a pointer to the word
                **
                ** Entry:
                **      ST(=sSTK)=0:
                **         D0 points to first letter of device word in memory
                **      ST(=sSTK)=1:
                **         D1 points to first letter of device word on stack
                **         D[A] points to the end of the specifier
                **
                ** Exit:
                **      Carry clear:
                **        Device word in B[W], zero-filled, first letter in B[B]
                **        P=0, carry clear if no error
                **        D0/D1 @ next character
                **      Carry set:
                **        Error (P, C[0] are error code)
                **
                ** Calls:       NXTCHR,BAKCHR,UCRANG,RANGEN
                **
                ** Uses.......
                **  Exclusive:      B[W],           P
                **  Inclusive: A[A],B[W],C[A],D0,D1,P (sSTK=0: D0; sSTK=1: D1)
                **
                ** Stk lvls:   2 (UCRANG)
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------    --------------------------------
                **  11/04/82      NZ          Added documentation
                **
                ********************************************************************
                ********************************************************************
F71C8 AF1       =GETDVW B=0     W
F71CB 7AC2              GOSUB   Nxtchr          Read first character
F71CF 4E2               GOC     GETDV2          Should NEVER happen...
                *
                * First character MUST be in [A-Z] or [a-z]
                *
F71D2 7123      GETDV0  GOSUB   Ucrang          Convert to upper case&check [A-Z]
F71D6 432               GOC     GETDV-          Done (not in [A-Z])
F71D9 AE8       GETDV1  B=A     B               Copy to B[B]...
F71DC 815               BSRC                    ...rotate to B[15:14]...
F71DF 815               BSRC
F71E2 96D               ?B#0    B               ...and check if room for more
F71E5 31                GOYES   GETDVr          No room...done
F71E7 7EA2              GOSUB   Nxtchr          Get next character
F71EB 421               GOC     GETDV2          Done...justify it
F71EE 76D2              GOSUB   Rangen          Check if this is numeric...
F71F2 56E               GONC    GETDV1          ...yes...save it
F71F5 4CD               GOC     GETDV0          Go always (Check if in [A-Z])
                *-
                *-
```

```
F71F8 03   GETDVr  RTNCC               Return, carry clear
           *-
           *-
F71FA 70D2 GETDV-  GOSUB  Bakchr        Back up this character
F71FE 97D  GETDV2  ?B#0   W             If whole word is zero, Error
F7201 60           GOYES  GETDV3        Not zero...continue
F7203 20           P=     =eDSPEC       Bad device word
F7205 02           RTNSC
           *-
           *-
F7207 96D  GETDV3  ?B#0   B             If B[B] is non-zero, done
F720A EE           GOYES  GETDVr        Return, clear carry
           *
           * If blank-filling is desired, do LCASC \ \; B=C B here
           *
F720C 815          BSRC
F720F 815          BSRC
F7212 54F          GONC   GETDV3        Go always
           ********************************************************************
           ********************************************************************
           **
           ** Name:       PROCDW - Process device word
           **
           ** Category:   FILUTL
           **
           ** Purpose:
           **      Given a device word in B[W], figure out what it is
           **      (ASSIGN WORD, RESERVED WORD, NULL, LOOP, DEVICE ID)
           **
           ** Entry:
           **      B[W] contains the device word
           **
           ** Exit:
           **      P=0
           **      Carry set if sequence number is permissable after this
           **      Carry clear if sequence number is not permissable
           **
           ** Calls:      CHKAIO,ROMTYP,(PRDWsb)
           **
           ** Uses.......
           **  Exclusive:           C[W],P
           **  Inclusive: A[A],B[B],C[W],P
           **
           ** Stk lvls:   2 (CHKAIO)(ROMTYP)
           **
           ** Detail:
           **      Try in following order: ASSIGN WORD, RESERVED WORD,
           **        NULL,LOOP,(other=DEVICE ID)
           **
           ** History:
           **
           **    Date     Programmer          Modification
           **  --------   ----------   ---------------------------------
           **  04/28/83      NZ        Changed LOOP and NULL to check
           **                          all 8 characters
```

```
866                ** 11/04/82      NZ          Added documentation
867                **
868                ************************************************************
869                ************************************************************
870 F7215 8E00 =PROCDW GOSUBL =CHKAIO       Check if ASSIGNIO
          00
871 F721B 500          RTNNC                If carry clear, found it
872 F721E 8E00         GOSUBL =ROMTYP       Check if reserved word
          00
873                *
874                * Carry indicates whether found or not (If not, ID)
875                *
876 F7224 533          GONC   PRDW30        Found...return, set carry
877 F7227 AF2          C=0    W             Clear high nibbles of C first
878 F722A 37E4         LCASC  \LLUN\        Check if device type="NULL"
          55C4
          C4
879 F7234 7220         GOSUB  PRDWsb        (Check for match)
880                *
881                * If carry clear, this is "NULL"
882                *
883 F7238 3100         LC(2)  =Null         This is the "NULL" device?
884 F723C 500          RTNNC                If no carry, NULL
885 F723F 37C4         LCASC  \POOL\        Check if device type="LOOP"
          F4F4
          05
886 F7249 7D00         GOSUB  PRDWsb        (Check for match)
887 F724D 3100         LC(2)  =Loop
888 F7251 560          GONC   PRDW30        If no carry, this is LOOP
889 F7254 3100         LC(2)  =DevID        C[4:2] is zero
890 F7258 02   PRDW30  RTNSC
891                *-
892                *-
893 F725A 975  PRDWsb  ?B#C   W
894 F725D 20           GOYES  PRDWs1
895 F725F D2   PRDWs1  C=0    A
896 F7261 01           RTN
897                ************************************************************
898                ************************************************************
899                **
900                ** Name:       PROCLT - Process literal device spec
901                **
902                ** Category:   FILUTL
903                **
904                ** Purpose:
905                **      Given a pointer to a device spec in memory, process it
906                **
907                ** Entry:
908                **      D0 @ device spec
909                **
910                ** Exit:
911                **      Carry clear:
912                **        P=0
913                **        Device type/device id in B[X]/B[W]
914                **        IF device type="*", *, or "" THEN C[X]=0
```

```
915             **        ELSEIF address THEN C[X] is address+loop*1024
916             **        ELSEIF LOOP then C[X] is "9F"+loop*4096
917             **        ELSEIF NULL then C[B] is "7F"
918             **        ELSEIF volume label THEN C[X] is "5F"+loop*4096
919             **        ELSEIF device type THEN C[X] is "3F"+loop*4096
920             **        ELSEIF device ID THEN C[X] is "1F"+loop*4096
921             **      Carry set:
922             **        Error (P, C[0] are error code)
923             **
924             ** Calls:      NXTCHR,BAKCHR,GETDVW,PROCDW,SAVEAC,EXPEX+,
925             **             GHEXBT,GADRR+,RESTST,SAVE2C,RESTD1,REST2C
926             **
927             ** Uses.......
928             **  Exclusive: A,B,C,     R1,R2,      D0,   P
929             **  Inclusive: A,B,C,D,R0,R1,R2,R3,R4,D0,D1,P,STMTD1[3:0],STMTR1,
930             **             FUNCxx, all RAM available to FCNS
931             **
932             ** Stk lvls:   4 (EXPEX+ {saves a level on GOSUB stack first})
933             **
934             ** History:
935             **
936             **    Date     Programmer            Modification
937             **  --------   ----------    --------------------------------
938             **  09/28/83      NZ         Updated documentation
939             **  04/12/83      NZ         Fixed loop # processing
940             **  03/17/83      NZ         Changed to use STMTD1, not STMTR0
941             **  03/01/83      NZ         Reworked volume label code
942             **  02/07/83      NZ         Added status save in EXPEX+ call
943             **  11/04/82      NZ         Added documentation
944             **
945             ********************************************************************
946             ********************************************************************
947 F7263 7232 =PRTCLT GOSUB  Nxtchr
948             *
949             * Should have carry ONLY if next token is EOL (Error)
950             *
951 F7267 4D0          GOC    PRLT05
952 F726A 20           P=     0             (This P=0 is not needed-NXTCHR)
953 F726C 3100         LC(2)  =tCOLON
954 F7270 962          ?A=C   B             Is this a ":"?
955 F7273 60           GOYES  PRTCld        Yes...continue
956 F7275 6E31 PRLT05  GOTO   PRLTer        Error
957             *-
958             *-
959             *
960             * Process literal device spec
961             *
962 F7279 14A  PRTCld  A=DAT0 B             Read it directly (can be tSEMIC)
963 F727C 161          D0=D0+ 2             Skip it
964 F727F 3100         LC(2)  =tLITRL
965 F7283 962          ?A=C   B             Is this a literal?
966 F7286 60           GOYES  PRLT12        Yes...get device word
967 F7288 6470         GOTO   PRLT50        No...continue checking
968             *-
969             *-
```

```
  970                *
  971                * Literal device spec
  972                *
  973 F728C 783F PRLT12  GOSUB  GETDVW        Get device word
  974 F7290 400          RTNC                 Error
  975 F7293 7E7F         GOSUB  PROCDW        Process device word
  976 F7297 450          GOC    PRLT15        Sequence number IS acceptable
  977 F729A 5E5          GONC   PRLT9.        Go always...NOT acceptable
  978                *-
  979                *-
  980                *
  981                * Now save it, get sequence #
  982                *
  983 F729D 7221 PRLT15  GOSUB  SAVEAC        Save C[3:0] in STMTD1,B in STMTR1
  984                *
  985                * Process literal sequence number
  986                *
  987 F72A1 74F1         GOSUB  Nxtchr
  988 F72A5 453          GOC    PRLT25        No next character...exit
  989 F72A8 3100         LC(2)  =tCOLON
  990 F72AC 966          ?A#C   B
  991 F72AF 82           GOYES  PRLT20        Back up...not a sequence #
  992                *
  993                * Sequence # found
  994                *
  995 F72B1 7F12         GOSUB  Expex+        Get the type expression
  996 F72B5 76E1         GOSUB  Restst        Restore status bits
  997 F72B9 8E00         GOSUBL =GHEXBT       Get type (sequence) from RAM
             00
  998 F72BF 400          RTNC                 Error
  999                *
 1000                * Now B[A] is the sequence #
 1001                *
 1002 F72C2 CD           B=B-1  A             If carry, error
 1003 F72C4 4E0          GOC    PRLteR        Error (zero)
 1004 F72C7 21           P=     1
 1005 F72C9 90D          ?B#0   P
 1006 F72CC 70           GOYES  PRLteR        Error (too big)
 1007 F72CE 20           P=     0
 1008 F72D0 5C0          GONC   PRLT30        Go always
 1009               *-
 1010               *-
 1011 F72D3 20   PRLteR  P=     =eRANGE
 1012 F72D5 02           RTNSC
 1013               *-
 1014               *-
 1015 F72D7 73F1 PRLT20  GOSUB  Bakchr
 1016 F72DB D1   PRLT25  B=0    A             Put sequence # in B[A](=0)
 1017               *
 1018               * Now B[A] is sequence #
 1019               *
 1020 F72DD 133  PRLT30  AD1EX
 1021 F72E0 8E00         GOSUBL =RESTD1       Restore type/address...
             00
 1022 F72E6 133          AD1EX                ...to A[A]
```

```
1023                *
1024                * Now A[A] is type, B[B] is sequence #
1025                *
1026 F72E9 8E00             GOSUBL =REST2C       Restore acc/dev ID to C[W]
           00
1027 F72EF AFD              BCEX   W              Seq # to C[A], acc/dev ID to B[W]
1028                *
1029                * Now A[A] is type; B[W] is acc/dev ID; C[A] is seq #
1030                *
1031 F72F2 F2               CSL    A
1032 F72F4 F2               CSL    A              Sequence # in C[XS] now
1033 F72F6 AE6              C=A    B              Restore type
1034 F72F9 6C50 PRLT9.      GOTO   PRLT90         Check for loop spec now
1035                *-
1036                *-
1037 F72FD      PRLT50
1038                *
1039                * Not a literal...check for volume label
1040                *
1041 F72FD 3100             LC(2)  =tSEMIC
1042 F7301 966              ?A#C   B
1043 F7304 21               GOYES  PRLT60
1044                *
1045                * This is a volume label
1046                *
1047 F7306 7EBE             GOSUB  GETDVW         Get volume label (Get device word)
1048 F730A 400              RTNC                  If carry, error
1049 F730D D2               C=0    A
1050 F730F 3100             LC(2)  =VolLbl        Indicate volume label
1051 F7313 524              GONC   PRLT90         Go always...check for loop spec
1052                *-
1053                *-
1054 F7316 3100 PRLT60      LC(2)  =t%            Check if device type
1055 F731A 966              ?A#C   B
1056 F731D 71               GOYES  PRLT70         Not device type...check "*"
1057                *
1058                * Type...get it
1059                *
1060 F731F 71B1             GOSUB  Expex+         Get the type expression from RAM
1061 F7323 7871             GOSUB  Restst         Restore status bits
1062 F7327 8E00             GOSUBL =GHEXBT        Get HEX byte from RAM
           00
1063 F732D 400              RTNC                  Error
1064 F7330 6C6F             GOTO   PRLT15         Finish it up
1065                *-
1066                *-
1067 F7334 3100 PRLT70      LC(2)  =t*
1068 F7338 966              ?A#C   B
1069 F733B 60               GOYES  PRLT75
1070 F733D D2               C=0    A              This is "*"
1071 F733F 03               RTNCC
1072                *-
1073                *-
1074 F7341 7981 PRLT75      GOSUB  Bakchr         Back up to start of expression
1075                *
```

```
1076            * Address
1077            *
1078 F7345 7B81         GOSUB  Expex+        Get address expression from RAM
1079 F7349 7251         GOSUB  Reststt       Restore status bits
1080 F734D 8E00         GOSUBL =GADRR+       Get address from RAM
           00
1081 F7353 400          RTNC                 Carry indicates error state
1082            *
1083            * Entry point to check for literal loop spec
1084            *
1085 F7356 14A  PRLT90  A=DAT0 B             Read next character directly
1086            *
1087            * Before LC, save C[A] on RSTK (C[A] is device spec info)
1088            *
1089 F7359 06           RSTK=C               Save C[A] on RSTK
1090 F735B 3100         LC(2)  =tSEMIC       Is it a tSEMIC (loop number)?
1091 F735F 966          ?A#C   B
1092 F7362 20           GOYES  PRLT95        Exit after restoring C
1093 F7364 07   PRLT95  C=RSTK               Restore C (if carry, done!)
1094 F7366 415          GOC    PRLTex        Exit (Done)
1095 F7369 161          D0=D0+ 2             Skip the tSEMIC
1096            *
1097            * Need to save B and C from EXPEX+
1098            *
1099 F736C 7350         GOSUB  SAVEAC        Save C[3:0] in STMTD1,B in STMTR1
1100            *
1101            * Process literal loop spec
1102            *
1103 F7370 7061         GOSUB  Expex+        Get loop # expression from RAM
1104 F7374 7721         GOSUB  Reststt       Restore status bits
1105 F7378 8E00         GOSUBL =GHEXBT       Get HEX byte from RAM
           00
1106 F737E 400          RTNC                 Error
1107            *
1108            * Now B[A] is the loop # + 1
1109            *
1110 F7381 3130         LC(2)  3
1111 F7385 9E1          ?B>C   B
1112 F7388 90           GOYES  PRLLer        Error...too big
1113 F738A D9           C=B    A             Return loop # in C[0]
1114 F738C CE           C=C-1  A             Offset for zero-based count
1115 F738E 560          GONC   PRLTxx        If carry, zero (error-too small)
1116 F7391 20   PRLLer  P=     =eRANGE
1117 F7393 02           RTNSC
1118            *-
1119            *-
1120 F7395 10A  PRLTxx  R2=C                 Save loop # in R2
1121 F7398 137          CD1EX
1122 F739B 8E00         GOSUBL =RESTD1       Restore type/address
           00
1123 F73A1 137          CD1EX
1124 F73A4 109          R1=C                 Type in R1
1125 F73A7 8E00         GOSUBL =REST2C
           00
1126 F73AD 12A          CR2EX                Device ID in R2, loop # in C[0]
```

```
1127 F73B0 669C         GOTO   PRSTEX        Finish it up
1128            *-
1129            *-
1130 F73B4 20   PRLIer  P=     =eDSPEC       Device spec error
1131 F73B6 02           RTNSC
1132            *-
1133            *-
1134 F73B8 10A  PRLIex  R2=C                  Save C[W] in R2...
1135 F73BB AF9          C=B    W              Put B[W] into R2[W] also
1136 F73BE 12A          CR2EX                 ...restore C[W], set R2=B[W]
1137 F73C1 03           RTNCC
1138            *-
1139            *-
1140 F73C3      SAVEAC
1141            *
1142            * Preserve STMTD1[4]
1143            *
1144 F73C3 137          CD1EX                 Save C[A] in D1
1145 F73C6 06           RSTK=C                Save D1 on RSTK
1146 F73C8 137          CD1EX                 Restore C[A]
1147 F73CB 1F00         D1=(5) =STMTD1
          000
1148 F73D2 15D3         DAT1=C 4              Write out the low 4 nibs ONLY
1149 F73D6 07           C=RSTK                Restore D1 from RSTK...
1150 F73D8 135          D1=C                  ...done
1151 F73DB AF9          C=B    W
1152 F73DE 8C00         GOLONG =SAVE2C        Save B[W] in STMTR1
          00
1153            ********************************************************************
1154            ********************************************************************
1155            **
1156            ** Name:       FXQPIL - Get a file name from memory (file spec)
1157            **
1158            ** Category:   FILUTL
1159            **
1160            ** Purpose:
1161            **      Fetch a filename from program memory
1162            **
1163            ** Entry:
1164            **      Exit conditions from GETSTR
1165            **      (ST[sSTK]=0: literal in memory, =1:string on stack)
1166            **      (P=0)
1167            **
1168            ** Exit:
1169            **      D0/D1 set to first non-character item
1170            **      Carry clear (filename found):
1171            **        R0[W] is the first 8 chars, A[3:0] the last 2
1172            **        (Both are blank-filled)
1173            **      Carry set (no filename found):
1174            **        A,R0 are zeroed
1175            **
1176            ** Calls:      FXQPnn,FXQPn+
1177            **
1178            ** Uses.......
1179            **  Exclusive: A[W],     C[W],R0,      P
```

```
1180            **  Inclusive: A[W],B[W],C[W],R0,D0,D1,P
1181            **
1182            ** Stk lvls:   3 (FXQPnn)
1183            **
1184            ** Algorithm:
1185            **      Check if literal and no file name; if so, return zero
1186            **      Get the first 8 chars; put in R0; if reached end, set
1187            **        A[3:0]=\  \, return
1188            **      Get last 2 chars; put in A[3:0]; return
1189            **
1190            ** History:
1191            **
1192            **    Date      Programmer            Modification
1193            **  --------    ----------    ---------------------------------
1194            **  11/04/82      NZ          Added documentation
1195            **
1196            ****************************************************************
1197            ****************************************************************
1198 F73E4 AF2  =FXQPIL C=0    W
1199 F73E7 108         R0=C                  Preclear file name (for null str)
1200 F73EA 860         ?ST=0  =sSTK          String expression?
1201 F73ED 70          GOYES  FXQP30         No...literal
1202            *
1203            * Check if this is a null string...if so, return
1204            *
1205 F73EF 8A8         ?A=0   A
1206 F73F2 33          GOYES  FXQP50         Null string
1207            *
1208            * Now get the characters of the name until not in [A-Z]
1209            * or string length exhausted (Build the string in B[W])
1210            *
1211            * This is also the entry point for reading from program memory
1212            *
1213 F73F4 2F   FXQP30  P=     15
1214 F73F6 308         LC(1)  8              C[S] is character counter
1215 F73F9 7E20        GOSUB  FXQPnn         Get the name until B is full
1216            *                               or END is reached or bad char
1217 F73FD AF4         A=B    W
1218 F7400 100         R0=A
1219 F7403 4F0         GOC    FXQP40         Carry if END or bad char
1220            *
1221            * A[B],B[W] contain first 8 chars...copy to R0
1222            *
1223 F7406 2F          P=     15
1224 F7408 302         LC(1)  2              Two more characters MAX
1225 F740B 7F10        GOSUB  FXQPn+         Get the last 2 chars of name
1226 F740F D4          A=B    A              Copy characters to A[3:0]
1227            *
1228            * Have a FULL filename now! (Next char better be ":")
1229            * (D1 is at next character)
1230            *
1231 F7411 03          RTNCC                 Return with it all set up
1232            *-
1233            *-
1234 F7413      FXQP40
```

```
1235                *
1236                * Filename with less than 8 chars in A[W], B[W]
1237                *
1238 F7413 3302             LCASC  \  \
          02
1239 F7419 DA              A=C    A              Set last 2 characters to blanks
1240 F741B 118             C=R0                  Get back first 8 chars to test
1241 F741E 8AA             ?C=0   A
1242 F7421 40              GOYES  FXQP50         Yes...zero it all
1243 F7423 03              RTNCC                 Next character @ D1
1244                *-
1245                *-
1246 F7425         FXQP50
1247                *
1248                * No chars in name...set full name equal to zero
1249                *
1250 F7425 D0              A=0    A              Clear the last 2 chars
1251 F7427 20              P=     =eDSPEC        Bad device spec
1252 F7429 02              RTNSC
1253                ****************************************************************
1254                ****************************************************************
1255                **
1256                ** Name:       FXQPnm - Read chars from memory/stack (count)
1257                **
1258                ** Category:   FILUTL
1259                **
1260                ** Purpose:
1261                **      Read characters from either the stack or program
1262                **      memory until either a count is exceeded or an end is
1263                **      reached
1264                **
1265                ** Entry:
1266                **      C[S] is byte count
1267                **      sSTK is set for STACK, clear for literal
1268                **      If ST[=sSTK]=1, D1 points to string, D[A] is end
1269                **      If ST[=sSTK]=0, D0 points to the literal
1270                **
1271                ** Exit:
1272                **      B[W] contains the filename (IF sFirst=1 AND bad char,B=0)
1273                **      Carry set if reached END or bad char, clear if count
1274                **      D0/D1 set to first character not used
1275                **      A[S] is the original byte count
1276                **      P=0
1277                **
1278                ** Calls:      NXTCHR,BAKCHR,UCRANG,RANGEN,BLANKC
1279                **
1280                ** Uses.......
1281                **  Exclusive: A[X],B[W],C[W],      P,ST[sFirst]
1282                **  Inclusive: A[W],B[W],C[W],D0,D1,P,ST[sFirst]
1283                **
1284                ** Stk lvls:   2 (UCRANG)
1285                **
1286                ** Detail:
1287                **      Reads characters until either:
1288                **      1) Count is reached
```

```
1289                **      2) A character NOT in [A-Z] is found
1290                **
1291                ** History:
1292                **
1293                **    Date      Programmer            Modification
1294                **   --------   ----------   --------------------------------
1295                **  04/29/83       NZ        Changed GOC after NXTCHR @ FXQPn1
1296                **                           to skip the BAKCHR @ FXQPn3
1297                **  03/19/83       NZ        Changed FXQPnm and FXQPn+ so that
1298                **                           FXQPnm sets =sFirst, FXQPn+ does
1299                **                           not change =sFirst
1300                **  11/04/82       NZ        Added documentation
1301                **  01/20/83       NZ        Added check for sFirst AND bad ch
1302                **
1303                ****************************************************************
1304                ****************************************************************
1305 F742B 850      =FXQPnm ST=1   =sFirst       Entry for first char
1306 F742E ACA      =FXQPn+ A=C    S             Save count in A[S]
1307 F7431 8E00           GOSUBL =BLANKC       Initially blanks
             00
1308 F7437 AF5            B=C    W
1309 F743A AC6            C=A    S             Use count in C[S], Save in A[S]
1310 F743D 7850 FXQPn1  GOSUB  Nxtchr        Get next character in A[B]
1311 F7441 433            GOC    FXQPn-        END
1312 F7444 7FA0           GOSUB  Ucrang        Convert to upper case
1313 F7448 5E0            GONC   FXQPn2        If carry clear, IS in [A-Z]
1314                *
1315                * Character not in [A-Z]...if this is First, Error
1316                *
1317 F744B 870            ?ST=1  =sFirst
1318 F744E 32             GOYES  FXQPn3        Error! (Bad first character)
1319 F7450 7470           GOSUB  Rangen        Check if this is a digit
1320 F7454 4C1            GOC    FXQPn3        Not a digit...error
1321                *
1322                * Have a valid character here
1323                *
1324 F7457 840  FXQPn2  ST=0   =sFirst       Clear for later chars
1325 F745A AE8            B=A    B             Save in B[B]...
1326 F745D 815            BSRC
1327 F7460 815            BSRC                 Rotate the character to B[15:14]
1328 F7463 A4E            C=C-1  S             Do more?
1329 F7466 94E            ?C#0   S
1330 F7469 4D             GOYES  FXQPn1        Yes...loop back
1331                *
1332                * Count reached
1333                *
1334                * Use A[XS] to indicate carry/no carry on exit
1335                *
1336 F746B AA0            A=0    XS
1337 F746E 541            GONC   FXQPn4        Go always
1338                *-
1339                *-
1340 F7471      FXQPn3
1341                *
1342                * Reached END/bad char
```

```
1343                *
1344 F7471 7950            GOSUB  Bakchr        Back up to last character
1345 F7475 AA0  FXQPn-     A=0    XS
1346 F7478 A2C             A=A-1  XS            Set A[XS]="F"
1347 F747B 860             ?ST=0  =sFirst       Is this the first char?
1348 F747E 50              GOYES  FXQPn4        No...continue
1349 F7480 AF1             B=0    W             Yes...set B[W]=0
1350 F7483 942  FXQPn4     ?A=C   S
1351 F7486 E0              GOYES  FXQPn5        Done
1352 F7488 811             BSLC
1353 F748B 811             BSLC
1354 F748E B46             C=C+1  S
1355 F7491 51F             GONC   FXQPn4        Go always
1356                *-
1357                *-
1358 F7494 B24  FXQPn5     A=A+1  XS            Set carry with A[XS]
1359 F7497 01              RTN
1360                *-
1361                *-
1362 F7499 8C00 Nxtchr     GOLONG =NXTCHR
           00
1363                *-
1364                *-
1365 F749F 8E00 Reststt    GOSUBL =TSAVDO       Save DO in function scratch
           00
1366 F74A5 8F00            GOSBVL =POPUPD       Pop GOSUB stack into D[A]
           000
1367 F74AC 07              C=RSTK
1368 F74AE DF              CDEX   A
1369 F74B0 06              RSTK=C               Restore second level
1370 F74B2 DB              C=D    A
1371 F74B4 06              RSTK=C               Restore calling level
1372 F74B6 8E00            GOSUBL =D1@AVE       Set D1 at AVMEME
           00
1373 F74BC 8E00            GOSUBL =TRESDO       Restore DO from function scratch
           00
1374 F74C2 8C00            GOLONG =RESTST
           00
1375                *-
1376                *-
1377 F74C8 8C00 Rangen     GOLONG =RANGEN
           00
1378                *-
1379                *-
1380 F74CE 8C00 Bakchr     GOLONG =BAKCHR
           00
1381                *-
1382                *-
1383 F74D4 07   Expex+     C=RSTK               Save calling level in A[A]
1384 F74D6 DA              A=C    A
1385 F74D8 07              C=RSTK               Pop second level to C[A]
1386 F74DA DE              ACEX   A             Second level to A[A], call to C[A]
1387 F74DC 06              RSTK=C               Push calling level back on stack
1388 F74DE 8E00            GOSUBL =TSAVDO       Save DO first
           00
```

```
1389 F74E4 8F00          GOSBVL =PSHMCR       Push microcode return on GOSUB
           000
1390 F74EB 8E00          GOSUBL =TRESDO       Restore DO
           00
1391 F74F1 8C00          GOLONG =EXPEX+
           00
1392            *-
1393            *-
1394 F74F7 8C00 =Ucrang GOLONG =UCRANG
           00
1395 F74FD               END
```

```
 ASLC12  Ext                  -   241
 ASRC4   Ext                  -   187
 BAKCHR  Ext                  -  1380
 BLANKC  Ext                  -  1307
 Bakchr  Abs 1012942 #F74CE   -  1380    98   315   367   408   483   570   661
                                 703   717   814  1015  1074  1344
 CHKAIO  Ext                  -   870
 CSRC12  Ext                  -   222
 CSRC2   Ext                  -   670
 D1=AVS  Ext                  -   206
 D1@AVE  Ext                  -   211   236  1372
 DTOH    Ext                  -   572   666   724
 DevID   Ext                  -   889
 DevTyp  Ext                  -   583
 EXPEX+  Ext                  -  1391
 Expex+  Abs 1012948 #F74D4   -  1383   995  1060  1078  1103
 FXQP30  Abs 1012724 #F73F4   -  1213  1201
 FXQP40  Abs 1012755 #F7413   -  1234  1219
 FXQP50  Abs 1012773 #F7425   -  1246  1206  1242
=FXQPIL  Abs 1012708 #F73E4   -  1198   176
=FXQPn+  Abs 1012782 #F742E   -  1306  1225
 FXQPn-  Abs 1012853 #F7475   -  1345  1311
 FXQPn1  Abs 1012797 #F743D   -  1310  1330
 FXQPn2  Abs 1012823 #F7457   -  1324  1313
 FXQPn3  Abs 1012849 #F7471   -  1340  1318  1320
 FXQPn4  Abs 1012867 #F7483   -  1350  1337  1348  1355
 FXQPn5  Abs 1012884 #F7494   -  1358  1351
=FXQPnm  Abs 1012779 #F742B   -  1305  1215
 GADRR+  Ext                  -  1080
 GADRS.  Abs 1012005 #F7125   -   665   642   659
 GADRS1  Abs 1011964 #F70FC   -   641   648
 GADRS2  Abs 1011992 #F7118   -   653   644
 GADRS3  Abs 1012027 #F713B   -   677   684
 GADRS4  Abs 1012099 #F7183   -   704   678   714
 GADRS5  Abs 1012109 #F718D   -   717   691
 GADRS6  Abs 1012113 #F7191   -   718   689   699   712
=GADRST  Abs 1011961 #F70F9   -   640   409
 GADRSb  Abs 1012095 #F717F   -   703   680
 GADRSo  Abs 1011988 #F7114   -   649   697
 GADRSs  Abs 1012076 #F716C   -   696   694   701   731   735   737
 GETDCK  Abs 1011304 #F6E68   -   107   100
 GETDI+  Abs 1011258 #F6E3A   -    85   200
 GETDI0  Abs 1011292 #F6E5C   -    99    92
 GETDI1  Abs 1011250 #F6E32   -    77    67
 GETDI2  Abs 1011300 #F6E64   -   103    95
 GETDI3  Abs 1011325 #F6E7D   -   114    86    97   122
 GETDI4  Abs 1011329 #F6E81   -   118    83
 GETDI5  Abs 1011331 #F6E83   -   119    73   110   113   230
=GETDID  Abs 1011225 #F6E19   -    64
=GETDIX  Abs 1011255 #F6E37   -    84
 GETDV-  Abs 1012218 #F71FA   -   814   798
 GETDV0  Abs 1012178 #F71D2   -   797   808
 GETDV1  Abs 1012185 #F71D9   -   799   807
 GETDV2  Abs 1012222 #F71FE   -   815   793   805
 GETDV3  Abs 1012231 #F7207   -   821   816   828
```

```
=GETDVW  Abs 1012168 #F71C8 -   791   316   416   973  1047
 GETDVr  Abs 1012216 #F71F8 -   811   803   822
=GETPI+  Abs 1011369 #F6EA9 -   176
 GETPI1  Abs 1011407 #F6ECF -   206   198
=GETPIL  Abs 1011360 #F6EA0 -   174
 GETPIn  Abs 1011528 #F6F48 -   253   215
 GETSTR  Ext                -    64   174
 GHEXBT  Ext                -   997  1062  1105
 GTYPS.  Abs 1011913 #F70C9 -   560   558
 GTYPS1  Abs 1011851 #F708B -   528   538
 GTYPS2  Abs 1011875 #F70A3 -   542   561   579
 GTYPS3  Abs 1011879 #F70A7 -   546   531
 GTYPS4  Abs 1011932 #F70DC -   570   548   555
 GTYPS5  Abs 1011936 #F70E0 -   571   529   553   566
=GTYPST  Abs 1011848 #F7088 -   527   341   399   437
 GTYPSd  Abs 1011918 #F70CE -   565   568
 Loop    Ext                -   887
 NXTCHR  Ext                -  1362
 Null    Ext                -   883
 Nxtchr  Abs 1012889 #F7499 -  1362    85   108   306   332   343   427   528
                               552   565   641   677   688   698   792   804
                               947   987  1310
 POPUPD  Ext                - 1366
 PRDW30  Abs 1012312 #F7258 -   890   876   888
 PRDWs1  Abs 1012319 #F725F -   895   894
 PRDWsb  Abs 1012314 #F725A -   893   879   886
 PRLLer  Abs 1012625 #F7391 -  1116  1112
 PRLT05  Abs 1012341 #F7275 -   956   951
 PRLT12  Abs 1012364 #F728C -   973   966
 PRLT15  Abs 1012381 #F729D -   983   976  1064
 PRLT20  Abs 1012439 #F72D7 -  1015   991
 PRLT25  Abs 1012443 #F72DB -  1016   988
 PRLT30  Abs 1012445 #F72DD -  1020  1008
 PRLT50  Abs 1012477 #F72FD -  1037   967
 PRLT60  Abs 1012502 #F7316 -  1054  1043
 PRLT70  Abs 1012532 #F7334 -  1067  1056
 PRLT75  Abs 1012545 #F7341 -  1074  1069
 PRLT9.  Abs 1012473 #F72F9 -  1034   977
 PRLT90  Abs 1012566 #F7356 -  1085  1034  1051
 PRLT95  Abs 1012580 #F7364 -  1093  1092
 PRLTer  Abs 1012660 #F73B4 -  1130   956
 PRLTex  Abs 1012664 #F73B8 -  1134  1094
 PRLTxx  Abs 1012629 #F7395 -  1120  1115
 PRLteR  Abs 1012435 #F72D3 -  1011  1003  1006
=PROCDW  Abs 1012245 #F7215 -   870   318   975
=PROCLT  Abs 1012323 #F7263 -   947    71   227
=PROCST  Abs 1011536 #F6F50 -   305    99
 PROCeX  Abs 1011833 #F7079 -   483   431
 PROCex  Abs 1011837 #F707D -   484   428
 PROCld  Abs 1012345 #F7279 -   962   955
 PROCna  Abs 1011826 #F7072 -   478   468
 PRST10  Abs 1011572 #F6F74 -   326   319   401
 PRST20  Abs 1011640 #F6FB8 -   367   337
 PRST25  Abs 1011644 #F6FBC -   368   333
 PRST27  Abs 1011646 #F6FBE -   372   361
```

```
 PRST30  Abs 1011669 #F6FD5 -   383   309
 PRST40  Abs 1011683 #F6FE3 -   393   385
 PRST50  Abs 1011703 #F6FF7 -   404   395
 PRST90  Abs 1011728 #F7010 -   423   320   380   390   400   410
 PRSTEX  Abs 1011783 #F7047 -   458   447  1127
 PRSTeR  Abs 1011636 #F6FB4 -   363   358
 PRSTed  Abs 1011532 #F6F4C -   301   307   317   344   350
 PRSTer  Abs 1011779 #F7043 -   451   444
 PRSTvl  Abs 1011715 #F7003 -   416   103
 PSHMCR  Ext                -  1389
 RANGEN  Ext                -  1377
 REST2C  Ext                -  1026  1125
 RESTD1  Ext                -  1021  1122
 RESTST  Ext                -  1374
 ROMTYP  Ext                -   872
 Rangen  Abs 1012936 #F74C8 -  1377   530   554   567   643   679   690   700
                               806  1319
 Restst  Abs 1012895 #F749F -  1365   996  1061  1079  1104
 SAVE2C  Ext                -  1152
 SAVEAC  Abs 1012675 #F73C3 -  1140   983  1099
 SETUP   Ext                -   125
 START   Ext                -   123
 STMTD1  Ext                -  1147
 TRESD0  Ext                -  1373  1390
 TSAVD0  Ext                -   119  1365  1388
 TermRq  Abs       0 #00000 -    62    84    96   199
 UCRANG  Ext                -  1394
=Ucrang  Abs 1012983 #F74F7 -  1394   308   797  1312
 VolLbl  Ext                -   419  1050
 eDSPEC  Ext                -   114   301   817  1130  1251
 eNORAM  Ext                -   253
 eRANGE  Ext                -   363   451   542   649  1011  1116
 sDevOK  Ext                -   120   196
 sFirst  Ext                -  1305  1317  1324  1347
 sSTK    Ext                -    66   197  1200
 t%      Ext                -  1054
 t*      Ext                -  1067
 tCOLON  Ext                -   953   989
 tLITRL  Ext                -   964
 tSEMIC  Ext                -  1041  1090
```

Input Parameters

  Source file name is NZ&FXQ::MS

  Listing file name is NZ/FXQ:TI:ML::-1

  Object file name is NZ%FXQ:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
                *
                *       N   N  ZZZZZ   &     PPPP     A    RRRR
                *       N   N      Z  & &    P   P   A A   R   R
                *       NN  N     Z   & &    P   P  A   A  R   R
                *       N N N    Z     &     PPPP   A   A  RRRR
                *       N  NN   Z     & & &  P      AAAAA  R R
                *       N   N  Z      &  &   P      A   A  R  R
                *       N   N  ZZZZZ   && &  P      A   A  R   R
                *
                        TITLE  NZ'S PARSE ROUTINES <831128.2333>
F74FD           *       ABS    #F74FD        TI%HP6 address (fixed)
                *
                * Status bits for Parse routines
                *
                * Global (BASIC System)
                *
                =InvalE EQU    0             Invalid expression if set
                =Digit  EQU    1             Digit found (CATCHR)
                =SpChar EQU    2             Special char found (CATCHR)
                =NumExp EQU    3             Numeric expression if set
                *
                * LOCAL (Used only in HPIL)
                *
                * ST(10) MUST be clear for any error exits! (Implied LET error)
                *
                =StarOK EQU    10            "*" OK (in device parse)
                =StrOK  EQU    StarOK        String OK (FRAme SPec parse)
                =ExprOK EQU    8             Expression OK (SEND parse)
                =EolOK  EQU    9             EOL OK (in SEND parse)
                =OptDev EQU    8             Device Spec is optional (Dev parse)
                ********************************************************************
                ********************************************************************
                **
                ** Name:       PRNTSp - Parse the PRINTER IS statement
                **
                ** Category:   STPARS
                **
                ** Purpose:
                **      Parse the PRINTER IS (and DISPLAY IS) statement
                **
                ** Entry:
                **      D1 points to the ASCII character string
                **      D0 points to the location where the tokens go
                **      D[A] is the end of available memory
                **      P=0
                **
                ** Exit:
                **      D0 positioned past the last token output by this routine
                **      D1 positioned past the last character accepted
                **      P=0
                **      Exits through ERRORP if error
                **
                ** Calls:      NTOKEN,<DVCPy*>
                **
                ** Uses.......
```

```
                **  Inclusive: A,B,C,D[15:5],R0,R1,R2,D0,D1,P,ST[11,10,8,7,3:0],
                **             FUNCD0,PRMCNT[0]
                **
                ** Stk lvls:   5 (DVCPy*)
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **  --------   ----------   --------------------------------
                **  11/23/83      NZ        Added documentation
                **
                ********************************************************************
                ********************************************************************
F74FD 7F36 =PRNTSp GOSUB  Ntoken        Get next token
F7501 3100         LC(2)  =tIS          "IS" token
F7505 966          ?A#C   B             Was the next token "IS"?
F7508 36           GOYES  PRNTPE        No..."IS" missing...error
F750A 6CA4         GOTO   DVCPy*        Yes...device spec, "*" permitted
                ********************************************************************
                ********************************************************************
                **
                ** Name:       OUTPp - Parse the OUTPUT statement
                ** Name:       ENTERp - Parse the ENTER statement
                **
                ** Category:   STPARS
                **
                ** Purpose:
                **
                **
                ** Entry:
                **      D1 points to the ASCII character string
                **      D0 points to the location where the tokens go
                **      D[A] is the end of available memory
                **      P=0
                **
                ** Exit:
                **      D1 positioned past last token output by this routine
                **      D1 positioned past last character accepted
                **      P=0
                **      Exits through ERRORP if error
                **
                ** Calls:      DVCPn*,OUTpCK,OUTBYT,USINGp,<DISPP>,<READP5>
                **
                ** Uses.......
                **  Inclusive: A,B,C,D[15:5],R0-R2,D0,D1,P,ST[11,10,8,7,3:0],
                **             FUNCD0,PRMCNT[0]
                **
                ** Stk lvls:   6 (DVCPn*)
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **  --------   ----------   --------------------------------
                **  11/23/83      NZ        Added documentation
                **
```

```
111             ****************************************************************
112             ****************************************************************
113             *
114             * OUTPUT parse
115             *
116 F750E 7E94 =OUTPp  GOSUB  DVCPn*        Parse device, "*" not permitted
117 F7512 7130         GOSUB  OUTpCK        See what is following...
118 F7516 8D00         GOVLNG =DISPP        Continue with DISPLAY parse
          000
119             *-
120             *-
121             *
122             * ENTER parse
123             *
124 F751D 7F84 =ENTERp GOSUB  DVCPn*        Parse device, "*" not permitted
125 F7521 7220         GOSUB  OUTpCK        See what is following...
126 F7525 8F00         GOSBVL =USINGp       Try to parse USING
          000
127 F752C 450          GOC    ENTR10        Parsed USING...don't change D1
128 F752F 171          D1=D1+ 2             No USING...skip semicolon
129 F7532 3100 ENTR10  LC(2)  =tSEMIC       Output tSEMIC
130 F7536 7185         GOSUB  OUTBYT
131 F753A 858          ST=1   8
132 F753D 849          ST=0   9
133 F7540 8D00         GOVLNG =READP5
          000
134             *-
135             *-
136             *
137             * OUTPUT and ENTER share a common syntax for device spec; both
138             * must be followed by one of the following:
139             *  1. USING
140             *  2. Semicolon
141             *  3. End of line
142             *
143 F7547 75F5 OUTpCK  GOSUB  Ntoken        Get next token
144 F754B 3100         LC(2)  =tUSING
145 F754F 962          ?A=C   B             Is it tUSING?
146 F7552 D0           GOYES  chkOK         Yes...accept it
147 F7554 3100         LC(2)  =tSEMIC
148 F7558 962          ?A=C   B             Is it tSEMIC?
149 F755B 40           GOYES  chkOK         Yes...accept it
150             *
151             * Not USING or Semicolon; if not EOL, then excess chars
152             *
153 F755D 07           C=RSTK               Return to main parse driver
154             *
155 F755F 3100 chkOK   LC(2)  =t@           Output a t@ to terminate the
156 F7563 7455         GOSUB  OUTBYT          device specifier
157 F7567 63B5         GOTO   RESPTR        Restore the pointer (NTOKEN)
158             *-
159             *-
160 F756B 20   PRNTPE  P=     =eSYNTx       "IS" token missing
161 F756D 6051         GOTO   Errorp        Syntax error (restore pointer)
162             ****************************************************************
```

```
163                ****************************************************************
164                **
165                ** Name:       INITp - Parse the INITIALIZE statement
166                **
167                ** Category:   STPARS
168                **
169                ** Purpose:
170                **      Parse the INITIALIZE statement
171                **
172                ** Entry:
173                **      D1 points to the ASCII character string
174                **      D0 points to the location where the tokens go
175                **      D[A] is the end of available memory
176                **      P=0
177                **
178                ** Exit:
179                **      D0 positioned past last token output by this routine
180                **      D1 positioned past last character accepted
181                **      P=0
182                **      Exits through ERRORP if error
183                **
184                ** Calls:      CONWUC,FILSp,NTOKEN,?A=CM+,CKNUM,<RESPTR>,
185                **             <ERROR!>,<ERRORP>
186                **
187                ** Uses.......
188                **  Inclusive: A,B,C,D[15:5],R0-R4,D0,D1,P,ST[11,7,3:0],FUNCD0,
189                **             PRMCNT[0]
190                **
191                ** Stk lvls:   6 (FILSp)
192                **
193                ** History:
194                **
195                **    Date     Programmer              Modification
196                **  --------   ----------    ---------------------------------
197                **  11/28/83      NZ         Added documentation
198                **
199                ****************************************************************
200                ****************************************************************
201 F7571 7756 =INITp  GOSUB  CONWUC        Convert word to upper case
202 F7575 AF6          C=A    W
203 F7578 3594         LCASC  \EZI\         End of INITIAL(IZE) keyword
            A554
204 F7580 976          ?A#C   W
205 F7583 44           GOYES  INITp1        "IZE" missing - ERROR...
206 F7585 175          D1=D1+ 6             Skip IZE
207                *
208                * Now have "INITIALIZE"
209                *
210 F7588 7BC2         GOSUB  FILSp         Parse filespec (with string?)
211 F758C 580          GONC   INITP.        No error...continue
212 F758F 8C00 Error!  GOLONG =ERROR!       Error with FILSp
            00
213                *-
214                *-
215 F7595 831  INITP.  ?XM=0
```

```
 216 F7598 80          GOYES  INITPO        OK
 217 F759A 20   MSGPAR  P=     =eMSPAr       Missing parameter
 218 F759C 6121        GOTO   Errorp        Error
 219            *-
 220            *-
 221 F75A0 7C95 INITPO  GOSUB  Ntoken        Next TOKEN
 222 F75A4      INITP2
 223 F75A4 8E00        GOSUBL =?A=CM+
           00
 224 F75AA 5D0         GONC   INITPR        No comma token...rtn, carry clear
 225 F75AD 7D05        GOSUB  oUT1TK        Comma token...output it
 226            *
 227            * Entry for <XWORD> <numeric expression>
 228            *
 229 F75B1      =XWRD1p
 230 F75B1 72B4        GOSUB  CKNUM         Check numeric expression
 231 F75B5 4D0         GOC    INITPE        Error jump
 232 F75B8 6265 =INITPR GOTO   RESPTR        Restore parse pointer
 233            *-
 234            *-
 235            *
 236            * Entry for <XWORD> <Expr> [, <Expr>]
 237            *
 238 F75BC      =STANp+
 239 F75BC 77A4        GOSUB  CKNUM         Check numeric expression
 240 F75C0 53E         GONC   INITP2        Valid numeric...continue
 241 F75C3 6AF0 INITPE  GOTO   Errorp        Parse error
 242            *-
 243            *-
 244 F75C7 20   INITp1  P=     =eSYNTx       Syntax error (No IZE)
 245 F75C9 64F0        GOTO   Errorp        Parse error
 246            ****************************************************************
 247            ****************************************************************
 248            **
 249            ** Name:       STANDp - Parse the STANDBY statement
 250            **
 251            ** Category:   STPARS
 252            **
 253            ** Purpose:
 254            **      Parse the STANDBY statement
 255            **
 256            ** Entry:
 257            **      D1 points to the ASCII character string
 258            **      D0 points to the location where the tokens go
 259            **      D[A] is the end of available memory
 260            **      P=0
 261            **
 262            ** Exit:
 263            **      D0 positioned past last token output by this routine
 264            **      D1 positioned past last character accepted
 265            **      P=0
 266            **      Exits through ERRORP if error
 267            **
 268            ** Calls:      LOOP#p,WRDSCN,CKNUM,<RESPTR>
 269            **
```

```
270                ** Uses.......
271                **  Inclusive: A,B,C,D[15:5],R0-R3,D0,D1,P,ST[11,7,3:0],FUNCD0,
272                **             PRMCNT[0]
273                **
274                ** Stk lvls:   6 (LOOP#p)
275                **
276                ** History:
277                **
278                **    Date      Programmer              Modification
279                **  --------   ----------   ------------------------------
280                **  11/28/83      NZ         Added documentation
281                **
282                ****************************************************************
283                ****************************************************************
284 F75CD 7B61  =STANDp GOSUB  LOOP#p        Parse optional loop #
285 F75D1 7EB5         GOSUB  wrdscn        Check for ON/OFF
286 F75D5 00           CON(2) =tON
287 F75D7 7D3          REL(3) RTNCC         ON...done
288 F75DA 00           CON(2) =tOFF
289 F75DC 2D3          REL(3) RTNCC         OFF...done
290 F75DF 00           CON(2) 0             Neither ON nor OFF...get num expr
291 F75E1 7635         GOSUB  RESPTR        (Restore input pointer first)
292 F75E5 66DF         GOTO   STANp+        Parse 1 or 2 expressions
293                ****************************************************************
294                ****************************************************************
295                **
296                ** Name:       LOCALp - Parse the LOCAL [LOCKOUT] statement
297                **
298                ** Category:   STPARS
299                **
300                ** Purpose:
301                **      Parse the LOCAL or LOCAL LOCKOUT statement
302                **
303                ** Entry:
304                **      D1 points to the ASCII character string
305                **      D0 points to the location where the tokens go
306                **      D[A] is the end of available memory
307                **      P=0
308                **
309                ** Exit:
310                **      D0 positioned past last token output by this routine
311                **      D1 positioned past last character accepted
312                **      P=0
313                **      Exits through ERRORP if error
314                **
315                ** Calls:      NTOKEN,OUT3TK,SVDOD1,CKNUM,RSDOD1,RESPTR,
316                **             <CLEARp>,<OUTBYT>
317                **
318                ** Uses.......
319                **  Inclusive: A,B,C,D[15:5],R0-R3,D0,D1,P,ST[11,7,3:0],FUNCD0,
320                **             PRMCNT[0]
321                **
322                ** Stk lvls:   5 (CKNUM)(<CLEARp>)
323                **
324                ** History:
```

```
                **
                **    Date     Programmer          Modification
                **  --------   ----------   --------------------------------
                **  11/28/83      NZ        Added documentation
                **
                ****************************************************************
                ****************************************************************
F75E9           =LOCALp
F75E9 7355              GOSUB  Ntoken
F75ED AF6               C=A    W              Set high nibbles for compare
                ***
                *       LC(6)  (=tLOCKO)~(=LEXPIL)~(=tXWORD)
                *
F75F0 35                NIBHEX 35             LC(6)
F75F2 00                CON(2) =tXWORD        ...
F75F4 00                CON(2) =LEXPIL        ..
F75F6 00                CON(2) =tLOCKO        .
                *
                ***
F75F8 976               ?A#C   W              Is it LOCAL LOCKOUT?
F75FB F1                GOYES  LOCLp1         No...restore, use REMOTE parse
                *
                * This is LOCAL LOCKOUT...output the token, check for loop #
                *
F75FD 7EC4              GOSUB  oUT3TK         Output 3 byte token
F7601 7E84 Loopp        GOSUB  SVDOD1         Save D0, D1 in R2
F7605 7E54              GOSUB  CKNUM          Check if numeric expr follows
F7609 20                P=     0              Regardless of carry, want P=0
F760B 5CA               GONC   INITPR         If good expr, done after RESPTR
F760E 7894              GOSUB  RSDOD1         Restore D0, D1 from R2
                *
                * Not a loop expression...put out a tCOMMA instead
                *
F7612 3100              LC(2)  =tCOMMA
F7616 64A4              GOTO   OUTBYT         Don't restore D1 (already correct)
                *-
                *-
F761A 7DF4 LOCLp1       GOSUB  RESPTR         Restore token pointer
                *
                * Fall into CLEARp
                *
                ****************************************************************
                ****************************************************************
                **
                ** Name:       CLEARp - Parse the CLEAR statement
                ** Name:       REMOTp - Parse the REMOTE statement
                ** Name:       TRIGp - Parse the TRIGGER statement
                **
                ** Category:   STPARS
                **
                ** Purpose:
                **       Parse CLEAR/REMOTE/TRIGGER/LOCAL statement
                **
                ** Entry:
                **       D1 points to the ASCII character string
```

```
380             **      D0 points to the location where the tokens go
381             **      D[A] is the end of available memory
382             **      P=0
383             **
384             ** Exit:
385             **      D0 positioned past last token output by this routine
386             **      D1 positioned past last character accepted
387             **      P=0
388             **      Exits through ERRORP if error
389             **
390             ** Calls:      EXPPAR
391             **
392             ** Uses.......
393             **  Inclusive: A,B,C,D[15:5],R0,R1,D0,D1,P,ST[11,7,3:0],FUNCD0,
394             **             PRMCNT[0]
395             **
396             ** Stk lvls:
397             **
398             ** History:
399             **
400             **    Date     Programmer             Modification
401             **  --------   ----------    --------------------------------
402             **  11/28/83      NZ         Added documentation
403             **
404             ****************************************************************
405             ****************************************************************
406             *
407             * Code above falls into this routine
408             *
409 F761E       =CLEARp
410 F761E       =REMOTp
411 F761E       =TRIGp
412 F761E 858           ST=1    OptDev          Device spec not required
413 F7621 84A           ST=0    =StarOK         No "*" allowed
414 F7624 6893          GOTO    DVCSPc          Device address parse
415             ****************************************************************
416             ****************************************************************
417             **
418             ** Name:       RESETp - Parse the RESET HPIL statement
419             **
420             ** Category:   STPARS
421             **
422             ** Purpose:
423             **      Parse the RESET HPIL statement
424             **
425             ** Entry:
426             **      D1 points to the ASCII character string
427             **      D0 points to the location where the tokens go
428             **      D[A] is the end of available memory
429             **      P=0
430             **
431             ** Exit:
432             **      D0 positioned past last token output by this routine
433             **      D1 positioned past last character accepted
434             **      P=0
```

```
435                **      Exits through ERRORP if error
436                **
437                ** Calls:      BLANK,CONWUC,<Loopp>
438                **
439                ** Uses.......
440                **  Inclusive: A,B,C,D[15:5],R0-R3,D0,D1,P,ST[11,7,3:0],FUNCD0,
441                **             PRMCNT[0]
442                **
443                ** Stk lvls:   5 (<Loopp>)
444                **
445                ** History:
446                **
447                **    Date     Programmer            Modification
448                **  --------   ----------   --------------------------------
449                **  11/28/83      NZ        Added documentation
450                **
451                ********************************************************************
452                ********************************************************************
453 F7628 7EF4 =RESETp GOSUB  BLANK
454 F762C 7C95        GOSUB  CONWUC        Convert word to upper case
455 F7630 AF6         C=A    W             Copy upper nibs for compare
456 F7633 3784        LCASC  \LIPH\
           0594
           C4
457 F763D 976         ?A#C   W
458 F7640 A2          GOYES  Errorx
459                *
460                * HPIL...leave as HPIL "RESET"
461                *
462 F7642 177         D1=D1+ 8
463 F7645 5BB         GONC   Loopp         Go always...check for loop #
464                ********************************************************************
465                ********************************************************************
466                **
467                ** Name:       OFFp - Parse OFF INTR/OFF IO
468                **
469                ** Category:   STPARS
470                **
471                ** Purpose:
472                **      Parse the tokens following tOFF (HPIL) for INTR or IO
473                **
474                ** Entry:
475                **      D1 points to the ASCII character string
476                **      D0 points to the location where the tokens go
477                **      D[A] is the end of available memory
478                **      P=0
479                **
480                ** Exit:
481                **      D0 positioned past last token output by this routine
482                **      D1 positioned past last character accepted
483                **      P=0
484                **      Exits through REST* if error
485                **
486                ** Calls:      WRDSCN
487                **
```

```
                ** Uses.......
                **  Inclusive: A,B,C,R0,R1,R2,D0,D1,P,ST[11,3:0]
                **
                ** Stk lvls:   4 (WRDSCN)
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **  --------   ----------   --------------------------------
                **  11/28/83      NZ        Added documentation
                **
                ****************************************************************
                ****************************************************************
F7648 7745 =OFFIOp GOSUB  wrdscn
F764C 00           CON(2) =tXWORD
F764E 00           CON(2) =LEXPIL
F7650 00           CON(2) =tINTRR
F7652 420          REL(3) IOp20
F7655 00           CON(2) 00
                *
                * If is not OFF INTR try OFF IO...
                *
F7657 70C4         GOSUB  RESPTR
                *
                * Fall into IOp
                *
                ****************************************************************
                ****************************************************************
                **
                ** Name:       IOp - Parse "IO" token
                **
                ** Category:   PARUTL
                **
                ** Purpose:
                **      Accept the "IO" token from the input stream (used for
                **      OFF IO, RESTORE IO, ASSIGN IO)
                **
                ** Entry:
                **      D1 points to the ASCII character string
                **      D0 points to the location where the tokens go
                **      D[A] is the end of available memory
                **      P=0
                **
                ** Exit:
                **      D0 positioned past last token output by this routine
                **      D1 positioned past last character accepted
                **      P=0
                **      Exits through REST* if error
                **
                ** Calls:      WRDSCN,<REST*>
                **
                ** Uses.......
                **  Inclusive: A,B,C,R0,R1,R2,D0,D1,P,ST[11,3:0]
                **
                ** Stk lvls:   4 (WRDSCN)
```

```
                **
                ** History:
                **
                **    Date     Programmer          Modification
                **  --------   ----------   --------------------------------
                **  11/28/83      NZ        Added documentation
                **
                ********************************************************************
                ********************************************************************
                *
                * Code above falls into this routine
                *
F765B 7435      =IOp    GOSUB  wrdscn        Get next token
F765F 00                CON(2) =tXWORD
F7661 00                CON(2) =LEXPIL
F7663 00                CON(2) =tIO
F7665 E00               REL(3) IOp10
F7668 00                CON(2) 00
F766A 20        Errorx  P=     0
F766C 8D00      000             GOVLNG =REST*        Restart parse as if never matched
                *-
                *-
F7673 185       IOp10   D0=D0- 6             Return (Don't output the token)
F7676 03        IOp20   RTNCC
                ********************************************************************
                ********************************************************************
                **
                ** Name:       ONINTp - Parse the ON INTR GOTO/GOSUB statement
                **
                ** Category:   STPARS
                **
                ** Purpose:
                **      Parse the ON INTR GOTO/GOSUB statement
                **
                ** Entry:
                **      D1 points to the ASCII character string
                **      D0 points to the location where the tokens go
                **      D[A] is the end of available memory
                **      P=0
                **
                ** Exit:
                **      D0 positioned past last token output by this routine
                **      D1 positioned past last character accepted
                **      P=0
                **      Exits through REST* if error
                **
                ** Calls:      WRDSCN,NTOKEN,<REST*>
                **
                ** Uses.......
                **  Inclusive: A,B,C,R0,R1,R2,D0,D1,P,ST[11,3:0]
                **
                ** Stk lvls:   4 (WRDSCN)
                **
                ** History:
```

```
                **
                **    Date     Programmer              Modification
                **  --------   ----------    --------------------------------
                **  11/28/83      NZ         Added documentation
                **
                ****************************************************************
                ****************************************************************
F7678 7715 =ONINTp GOSUB  wrdscn
F767C 00           CON(2) =tXWORD
F767E 00           CON(2) =LEXPIL
F7680 00           CON(2) =tINTRR
F7682 900          REL(3) ONINp1
F7685 00           CON(2) 00
F7687 62EF         GOTO   Errorx
                *-
                *-
F768B 185  ONINp1  D0=D0- 6              Don't output the INTR token
F768E 7EA4         GOSUB  Ntoken
F7692 858          ST=1   8              Set ON ERROR flag (single branch)
F7695 8D00         GOVLNG =ONP40
      000
                ****************************************************************
                ****************************************************************
                **
                ** Name:       ASGNp - Parse the ASSIGN IO statement
                **
                ** Category:   STPARS
                **
                ** Purpose:
                **      Parse the ASSIGN IO statement
                **
                ** Entry:
                **      D1 points to the ASCII character string
                **      D0 points to the location where the tokens go
                **      D[A] is the end of available memory
                **      P=0
                **
                ** Exit:
                **      D0 positioned past last token output by this routine
                **      D1 positioned past last character accepted
                **      P=0
                **      Exits through ERRORP if error
                **
                ** Calls:      IOp,CKSTR,NTOKEN,OUTBYT,<RESPTR>,<ERRORP>
                **
                ** Uses.......
                **  Inclusive: A,B,C,D[15:5],R0,R1,R2,D0,D1,P,ST[11,7,3:0],
                **             FUNCD0,PRMCNT[0]
                **
                ** Stk lvls:   5 (CKSTR)(IOp)
                **
                ** History:
                **
                **    Date     Programmer              Modification
                **  --------   ----------    --------------------------------
```

```
                ** 11/28/83      NZ          Added documentation
                **
                ****************************************************************
                ****************************************************************
F769C 7B8F      =ASGNp  GOSUB   IOp          First check for "IO"
                *
                * If IOp returns, found "IO"
                *
F76A0 70E3              GOSUB   CKSTR        Check for valid string (carry=NO)
F76A4 5F1               GONC    ASGNp2       Valid...restore pointer, done
F76A7 7594              GOSUB   Ntoken       Get the token
F76AB 3100              LC(2)   =t*
F76AF 966               ?A#C    B
F76B2 A0                GOYES   ASGNp1       Error...illegal parameter
F76B4 7C05              GOSUB   OUT:          ASSIGN IO *...output the tCOLON,
F76B8 6204              GOTO    OUTBYT       Output the t*, return, carry clear
                *-
                *-
F76BC 20        ASGNp1  P=      =eILPAr      Illegal parameter
F76BE 8C00      Errorp  GOLONG  =ERRORP      Error...restore pointer, exit
      00
                *-
                *-
F76C4 6654      ASGNp2  GOTO    RESPTR
                ****************************************************************
                ****************************************************************
                **
                ** Name:        SENDp - Parse the SEND statement
                **
                ** Category:    STPARS
                **
                ** Purpose:
                **      Parse the SEND statement
                **
                ** Entry:
                **      D1 points to the ASCII character string
                **      D0 points to the location where the tokens go
                **      D[A] is the end of available memory
                **      P=0
                **
                ** Exit:
                **      D0 positioned past last token output by this routine
                **      D1 positioned past last character accepted
                **      P=0
                **      Exits through ERRORP if error
                **
                ** Calls:       LOOP#p,FRASPp,ST!NOp,?A=CM+,RESPTR,BLANK,CONWUC,
                **              OUTBYT,OUTNBS
                **
                ** Uses.......
                **  Inclusive: A,B,C,D[15:5],R0-R3,D0,D1,P,ST[11:7,3:0],FUNCD0,
                **              PRMCNT[0]
                **
                ** Stk lvls:    6 (LOOP#p)
                **
```

```
705             ** Algorithm:
706             **      SENDp: Parse optional loop #                    (LOOP#p)
707             **
708             **      SENDP1:Attempt to parse a frame spec            (FRASPp)
709             **             If successful frame spec, goto SENDP1
710             **
711             **             If expression is not permitted here, goto SENDP5
712             **             Attempt to parse a string or number      (ST!NOp)
713             **             If unsucessful, goto SENDP5
714             **
715             **      SENDP2:Check if a comma follows (more expr)     (?A=CM+)
716             **             If no comma, goto SENDP3 (check for EOL)
717             **             Attempt to parse a string or number      (ST!NOp)
718             **             If successful, goto SENDP2
719             **
720             **      SENDlp:While character is a blank, back up one char
721             **             Goto SENDP5
722             **
723             **      SENDP3:Restore input pointer                    (RESPTR)
724             **             Get next character                       (BLANK )
725             **
726             **             If EOL is permitted here, then
727             **               Read next 3 characters
728             **               If characters = "EOL" then output "EOL"
729             **               Get next character
730             **               endif
731             **
732             **      SENDP4:Attempt to parse a frame spec            (FRASPp)
733             **             If successful, goto SENDP1
734             **
735             **      SENDP5:Clear ST[10] (Implied LET flag)
736             **             RTNCC
737             **
738             ** History:
739             **
740             **    Date     Programmer             Modification
741             **  --------   ----------    --------------------------------
742             **  11/28/83      NZ         Updated documentation
743             **
744             ****************************************************************
745             ****************************************************************
746             *
747             * Syntax:
748             *   SEND [<loop #>;] { <keyword> [ <num expr> | <str expr> [ ,
749             *    <num expr> | <str expr> ]* ] }*
750             *
751             *      (num expr is not be allowed for some of the keywords)
752             *      (str expr is not be allowed for some of the keywords)
753             *
754             *   Definitions:
755             *      <keyword> ::= DATA | END | IDY | UNL | LISTEN | UNT |
756             *             TALK | SAD | DDL | DDT | RDY | IFC | LPD | GTL |
757             *             SDC | CMD | MLA | MTA
758             *      <num expr> ::= numeric expression
759             *      <str expr> ::= string expression
```

```
                *       <loop #> ::= numeric expression in the range [1,3]
                *
F76C8           =SENDp
                *
                * LOOP#p compiles either <nothing> or <tSEMIC><num expr><tCOMMA>
                * It also calls BLANK, leaving the next char in A[B]
                *
F76C8 7070              GOSUB  LOOP#p           Parse loop number, if any
                *
                * ST(8) (=ExprOK) is clear from the entry to SEND parse
                *
                * FRASPp compiles <tCOLON><text string>. If not a valid frame,
                * returns with DO restored, carry SET.
                * A[B] is the next item, D1 points to the next item
                * If carry is CLEAR, FRASPp sets/clears ST(StrOK), ST(EolOK).
                * If carry is SET, FRASPp does not alter ST(StrOK),ST(EolOK).
                *
F76CC 7990 SENDP1       GOSUB  FRASPp           Frame spec parse
F76D0 5BF               GONC   SENDP1           If valid frame spec, try another
                *
                * ST(ExprOK) indicates if an expression makes sense here. If
                * it is not set and FRASPp returned with carry set, this is
                * a parse error!! (Expression following a frame spec that does
                * not take an expression)
                *
F76D3 868               ?ST=0  =ExprOK          Does an expression make sense?
F76D6 16                GOYES  SENDP5           No...exit! (Anything else: error)
                ********
                *
                * ST!NOp compiles {<tCOMMA> followed by <str expr>|<num expr>}
                * if no error has been detected; a string expresion is
                * accepted only if ST(StrOK) is SET, else errors on string.
                * An EOL is accepted if and only if ST(EolOK) is true.
                * An expression is accepted if and only if ST(ExprOK) is true.
                * A[B] is next token on return from ST!NOp; carry indicates
                * status (Carry set=error; carry clear=accepted, compiled)
                *
F76D8 7051              GOSUB  ST!NOp           Parse initial string | number
F76DC 4A5               GOC    SENDP5           No expression specified...done
                *
                * One expression given...check if another expression follows
                *
F76DF 7000 SENDP2       GOSUB  =?A=CM+
F76E3 571               GONC   SENDP3           No comma follows...check EOL
                *
                * Found a comma...MUST find another expression!
                *
F76E6 7241              GOSUB  ST!NOp           Parse string | number
F76EA 54F               GONC   SENDP2           Valid...check for another!
                *
                * Didn't find a valid expression...back up to the comma
                *
                * (ST!NOp leaves C[B]=\ \)
                *
F76ED 1C1  SENDlp       D1=D1- 2
```

```
 815 F76F0 14B          A=DAT1 B
 816 F76F3 962          ?A=C   B
 817 F76F6 7F           GOYES  SENDlp
 818 F76F8 5E3          GONC   SENDP5        Go always
 819             *-
 820             *-
 821 F76FB 7C14 SENDP3  GOSUB  RESPTR        Restore pointer...
 822 F76FF 7724         GOSUB  BLANK         ...Skip blanks, read in character
 823 F7703 869          ?ST=0  =EolOK        Is EOL permitted here?
 824 F7706 A2           GOYES  SENDP4        No...continue
 825             *
 826             * Check if this is EOL (If so, output it and get next frame)
 827             *
 828 F7708 70C4         GOSUB  CONWUC        Convert to upper case
 829 F770C AF6          C=A    W             (To facilitate compare)
 830 F770F 3554         LCASC  \LOE\         EOL
           F4C4
 831 F7717 976          ?A#C   W
 832 F771A 61           GOYES  SENDP4        Not EOL...continue
 833 F771C 175          D1=D1+ 6             Skip EOL
 834 F771F 71A4         GOSUB  OUT:          Output 1 byte from C[B]
 835 F7723 AEE          ACEX   B
 836 F7726 25           P=     5
 837 F7728 7DA3         GOSUB  oUTNBS        Output 6 nibbles from A[5:0]
 838 F772C 7AF3         GOSUB  BLANK         Skip to next token
 839             *
 840             * If here, MUST have another frame spec, else error!
 841             *
 842 F7730 7530 SENDP4  GOSUB  FRASPp
 843 F7734 579          GONC   SENDP1        Found frame spec...continue
 844             *
 845             * NOT a frame spec...unrecognized type
 846             *
 847             * Fall through to return to parse driver
 848             *
 849 F7737 84A  SENDP5  ST=0   =StrOK        Clear this bit for LINE PARSE
 850 F773A 03           RTNCC
 851             ****************************************************************
 852             ****************************************************************
 853             **
 854             ** Name:       LOOP#p - Parse an optional HPIL loop specifier
 855             **
 856             ** Category:   PARUTL
 857             **
 858             ** Purpose:
 859             **       Parse an optional loop number...if one present, output
 860             **       the tokens for it
 861             **
 862             ** Exit:
 863             **       A[B] is next char, D1 points at next character
 864             **       If <loop #> found, compiled code generated
 865             **
 866             ** Entry:
 867             **       D1 points to the ASCII character string
 868             **       D0 points to the location where the tokens go
```

```
869             **      D[A] is the end of available memory
870             **      P=0
871             **
872             ** Exit:
873             **      A[B] is next character (at D1)
874             **      D0 positioned past last token output by this routine
875             **      D1 positioned past last character accepted
876             **      P=0
877             **      Carry clear
878             **
879             ** Calls:       SVD0D1,OUTBYT,CKNUM,OUT1TK,RSD0D1,BLANK
880             **
881             ** Uses.......
882             **  Inclusive: A,B,C,D[15:5],R0-R3,D0,D1,P,ST[11,7,3:0],FUNCD0,
883             **             PRMCNT[0]
884             **
885             ** Stk lvls:   5 (CKNUM)
886             **
887             ** History:
888             **
889             **    Date     Programmer             Modification
890             **  --------   ----------   --------------------------------
891             **  11/28/83      NZ        Updated documentation
892             **
893             ********************************************************************
894             ********************************************************************
895             *
896             * Syntax:
897             *   Input stream: [ <num expr> ; ]
898             *   Compiled code: [ <tSEMIC> <num expr> <tSEMIC> ]
899             *
900 F773C 7353 =LOOP#p GOSUB  SVD0D1        Save D0, D1
901 F7740 20          P=     0
902 F7742 3100        LC(2)  =tSEMIC
903 F7746 7173        GOSUB  OUTBYT        Output the semicolon in case OK
904 F774A 7913        GOSUB  CKNUM         Check numeric expression
905 F774E 421         GOC    LOOP#1        Not good...restore,nchar, return
906             *
907             * This was a valid numeric expression (B[B] is ntoken)
908             *
909             * Check for trailing semicolon...
910             *
911 F7751 3100        LC(2)  =tSEMIC
912 F7755 966         ?A#C   B
913 F7758 90          GOYES  LOOP#1        Not semicolon...don't accept!
914             *
915             * Output a trailing tSEMIC!
916             *
917 F775A 7063        GOSUB  oUT1TK        (tSEMIC in A[B] now)
918 F775E 560         GONC   LOOP#2        Go always...get next char
919             *-
920             *-
921 F7761       LOOP#1
922             *
923             * Restore D0, D1; then get next char
```

```
                *
F7761 7543      GOSUB  RSD0D1        Restore D0, D1
F7765 64C3 LOOP#2 GOTO  BLANK        Get next character
                ********************************************************************
                ********************************************************************
                **
                ** Name:       FRASPp - Parse an HPIL frame specifier
                **
                ** Category:   PARUTL
                **
                ** Purpose:
                **      Frame spec parse for HPIL frame descriptors
                **
                ** Entry:
                **      A[B] is next character (at D1)
                **      D1 points to the ASCII character string
                **      D0 points to the location where the tokens go
                **      D[A] is the end of available memory
                **      P=0
                **
                ** Exit:
                **      A[B] is next item (at D1)
                **      If carry set, not valid input (D0,D1 restored)
                **      If carry clear, output <tCOLON><text string>.
                **          ST(StrOK) is set if string OK next, clear if not
                **          ST(EolOK) is set if EOL is OK next, else clear
                **          ST(ExprOK) is set if expression makes sense next
                **      D0 positioned past last token output by this routine
                **      D1 positioned past last character accepted
                **      P=0
                **
                ** Calls:      UCRANG,OUTBYT,FRAMEE,OUTNBS,<BLANK>
                **
                ** Uses.......
                **  Inclusive: A,B,C,R0,R1,P
                **
                ** Stk lvls:   2 (UCRANG)(OUTBYT)(FRAMEE)(OUTNBS)
                **
                ** History:
                **
                **    Date     Programmer              Modification
                **  --------   ----------    --------------------------------
                **  11/28/83      NZ         Updated documentation
                **
                ********************************************************************
                ********************************************************************
                *
                * Syntax:
                *   Input stream: <alpha text string>
                *   Token output: <tCOLON> <validated text string>
                *
F7769 7000 =FRASPp GOSUB  =Ucrang    Check if valid input...
F776D 400       RTNC                 If carry, not valid input!
F7770 7054      GOSUB  OUT:          Output a tCOLON before frame spec
F7774 AEE       ACEX   B             (OUTBYT does ACEX B)
```

```
 979 F7777 133          AD1EX
 980 F777A 101          R1=A                Save input pointer in R1
 981 F777D 133          AD1EX               (A,D1 unchanged)
 982             *
 983             * A[B] is first character...continue until not in [A-Z]
 984             * D1 points at first character
 985             *
 986 F7780 AC2          C=0    S            Count in C[S]
 987 F7783 814  FRASP1  ASRC
 988 F7786 814          ASRC                Save characters in high nibbles
 989 F7789 B46          C=C+1  S
 990 F778C 171          D1=D1+ 2            Point to next character
 991 F778F 14B          A=DAT1 B
 992 F7792 7000         GOSUB  =Ucrang      Check if in [A-Z]
 993 F7796 5CE          GONC   FRASP1       Yes...continue
 994             *
 995             * Got a character NOT in [A-Z]...rotate text, check it
 996             *
 997 F7799 80DF         P=C    15           Use P for the character count!
 998 F779D 0D           P=P-1               Decrement for base zero carry
 999 F779F 810  FRASP2  ASLC
1000 F77A2 810          ASLC                Shift one character
1001 F77A5 0D           P=P-1
1002 F77A7 57F          GONC   FRASP2       If no carry, not done shifting
1003             *
1004             * Now A[W] is the text, C[S] is the length in bytes
1005             *
1006 F77AA A46          C=C+C  S
1007 F77AD A4E          C=C-1  S            Offset for zero-based count
1008 F77B0 80DF         P=C    15
1009 F77B4 A96          C=A    WP           C[W] is now set up for FRAMEE
1010 F77B7 108          R0=C                Save text in R0
1011 F77BA D0           A=0    A            Clear A[B] for FRAMEE
1012 F77BC 8E00         GOSUBL =FRAMEE
          00
1013 F77C2 4E5          GOC    FRASP3       Error...not a valid frame
1014             *
1015             * C[S] is the length of the frame
1016             *
1017             * Valid frame...write it out, return
1018             *
1019 F77C5 110          A=R0
1020 F77C8 ACA          A=C    S            Write out only the specified nibs
1021             *
1022             * Set D1 past the last ACCEPTED character
1023             *
1024 F77CB 80DF         P=C    15           Set P=WP value of length
1025 F77CF 119          C=R1                Set the input pointer past frame
1026 F77D2 809          C+P+1               Skip the chars
1027 F77D5 135          D1=C                Set D1 just past the characters
1028             *
1029 F77D8 AF6          C=A    W            Copy high nibbles of A[W] to C[W]
1030 F77DB 84A          ST=0   =StrOK       String NOT ok unless CMD/DATA
1031 F77DE 849          ST=0   =EolOK       EOL NOT ok except after DATA
1032 F77E1 848          ST=0   =ExprOK      Expression not OK unless mask#0
```

```
1033 F77E4 969          ?B=0   B
1034 F77E7 50           GOYES  FRASPx        Mask IS zero...expression not OK
1035 F77E9 858          ST=1   =ExprOK       Non-zero mask...expression OK
1036 F77EC 2F   FRASPx  P=     15
1037 F77EE 3653         LC(7)  (\DMC\)*16+5  C[S]=5, C[5:0]="CMD" (reversed)
          4D44
          4
1038 F77F7 972          ?A=C   W
1039 F77FA 51           GOYES  FRASPy        Match...StrOK
1040             *
1041             * Following instruction is too big for LC(x)
1042             *      LC(9)  (\ATAD\)*16+7 C[S]=7, C[7:0]="DATA" (reversed)
1043 F77FC 387          NIBHEX 387           LC(9)..7
1044 F77FF 4414         NIBASC \DATA\
          4514
1045             *
1046 F7807 976          ?A#C   W
1047 F780A 80           GOYES  FRASPn
1048 F780C 859          ST=1   =EolOK        EOL is OK here
1049 F780F 85A  FRASPy  ST=1   =StrOK        String expression OK here
1050 F7812 AC6  FRASPn  C=A    S
1051 F7815 80DF         P=C    15
1052 F7819 7CB2         GOSUB  oUTNBS        Output the nibbles in A[WP]
1053 F781D 6C03         GOTO   BLANK         Skip to next non-blank char
1054             *-
1055             *-
1056 F7821       FRASP3
1057             *
1058             * Restore D0, D1
1059             *
1060 F7821 181          D0=D0- 2             Back up over tCOLON
1061 F7824 119          C=R1                 Restore D1 (Input pointer)...
1062 F7827 135          D1=C                 ...from R1
1063 F782A 02           RTNSC                Return with carry SET (bad frame)
1064             ********************************************************************
1065             ********************************************************************
1066             **
1067             ** Name:       ST!NOp - Parse a string or numeric expression
1068             **
1069             ** Category:   PARUTL
1070             **
1071             ** Purpose:
1072             **      Parse either a string or numeric expression (String OK
1073             **      only if ST(StrOK) is set
1074             **
1075             ** Entry:
1076             **      D1 points to the ASCII character string
1077             **      D0 points to the location where the tokens go
1078             **      D[A] is the end of available memory
1079             **      P=0
1080             **
1081             ** Exit:
1082             **      Next token in A[B] if carry clear, next char if set
1083             **      Carry clear if accepted; <tCOMMA><expr> compiled
1084             **      Carry set if error; pointers restored
```

```
1085                ** DO positioned past last token output by this routine
1086                ** D1 positioned past last character accepted
1087                ** P=0
1088                **
1089                ** Calls:      SVDOD1,OUTBYT,EXPPAR,RSDOD1,BLANK
1090                **
1091                ** Uses.......
1092                **  Inclusive: A,B,C,D[15:5],R0-R2,D0,D1,P,ST[11,7,3:0],FUNCD0,
1093                **             PRMCNT[0]
1094                **
1095                ** Stk lvls:   4 (EXPPAR)
1096                **
1097                ** History:
1098                **
1099                **    Date     Programmer             Modification
1100                **  --------   ----------   ------------------------------
1101                **  11/28/83      NZ        Updated documentation
1102                **
1103                ********************************************************************
1104                ********************************************************************
1105                *
1106                * Syntax:
1107                *   Input stream: <num expr> | <str expr>
1108                *   Token output: <tCOMMA> <legal expr>
1109                *
1110 F782C          =ST!NOp
1111                *
1112                * First save D0,D1 in R2,R3
1113                *
1114 F782C 7362           GOSUB  SVDOD1        Save D0, D1
1115 F7830 3100           LC(2)  =tCOMMA
1116 F7834 7382           GOSUB  OUTBYT        Output the Comma token
1117 F7838 7422           GOSUB  Exppar        Check if expression
1118 F783C 870            ?ST=1  InvalE        Is it invalid?
1119 F783F E0             GOYES  ST!NO2        Invalid...restore
1120 F7841 873            ?ST=1  NumExp        Is it valid numeric?
1121 F7844 70             GOYES  ST!NO1        Yes...accept it!
1122                *
1123                * String...check if StrOK...if OK, accept; if not, restore
1124                *
1125 F7846 86A            ?ST=0  =StrOK
1126 F7849 40             GOYES  ST!NO2        Not OK...restore
1127 F784B          ST!NO1
1128                *
1129                * Accept it all now (The ntoken is in A[B])
1130                *
1131 F784B 03             RTNCC                Carry clear=accepted
1132                *-
1133                *-
1134 F784D          ST!NO2
1135                *
1136                * Not accepted...restore and return with next char in A[B]
1137                *
1138 F784D 7952           GOSUB  RSDOD1        Restore D0, D1
1139 F7851 75D2           GOSUB  BLANK         Skip blanks, read next character
```

```
1140 F7855 02          RTNSC                Return, carry SET
1141            ********************************************************************
1142            ********************************************************************
1143            **
1144            ** Name:       FILSPp - Parse an HPIL file specifier
1145            ** Name:       FILSp - Parse an HPIL file specifier (string OK)
1146            ** Name:       DEVSPp - Parse an HPIL device specifer (got :)
1147            ** Name:       DVSPp - Parse an HPIL device specifer (* OK)
1148            **
1149            ** Category:   PARUTL
1150            **
1151            ** Purpose:
1152            **      Routine to parse a file and/or device specifier
1153            **
1154            ** Entry:
1155            **      D1 points to the ASCII character string
1156            **      D0 points to the location where the tokens go
1157            **      D[A] is the end of available memory
1158            **      P=0
1159            **
1160            ** Exit:
1161            **      D0 positioned past last token output by this routine
1162            **      D1 positioned past last character accepted
1163            **      P=0
1164            **      Carry set if error (C[3:0] is error #)
1165            **          (D1 points at the erroneous item)
1166            **      Carry clear if OK (D1 points past file spec, A is next
1167            **          token, D0 is set properly,A[S]#0 if filename found)
1168            **
1169            ** Calls:      CKSTR,OUTBYT,NAMEpb,OUT2TC,NAMEp,NTOKEN,OUT1TK,
1170            **             CKNUM+,CKNUM-,RESPTR,SVDOD1,CATCH+,RSDOD1
1171            **
1172            ** Uses.......
1173            **  Inclusive: A,B,C,D[15:5],R0-R4,D0,D1,P,ST[11,10,7,3:0],
1174            **             FUNCD0,PRMCNT[0]
1175            **
1176            ** Stk lvls:   FILSPp: 5 (CKNUM)
1177            ** Stk lvls:   FILSp:  5 (CKSTR)(CKNUM)
1178            ** Stk lvls:   DEVSPp: 4 (CKNUM+)(NAMEp)
1179            ** Stk lvls:   DVSPp:  4 (CKNUM+)(NAMEp)
1180            **
1181            ** History:
1182            **
1183            **    Date     Programmer          Modification
1184            **  --------   ----------   --------------------------------
1185            **  11/28/83      NZ        Updated documentation
1186            **
1187            ********************************************************************
1188            ********************************************************************
1189            *
1190            * File specifier syntax:
1191            *   Input stream:
1192            *       <string expression>
1193            *   or  [ <file name> ] : <device specifier>
1194            *   or  [ <file name> ] . <volume label>
```

```
1195            *   Token output:
1196            *       <string expression>
1197            *   or  <tLITRL> [ <file name> ] <tCOLON> <device specifier>
1198            *   or  <tLITRL> [ <file name> ] <tSEMIC> <volume label>
1199            *
1200            * Device specifier syntax:
1201            *   Input stream:
1202            * 1)    <string expression>                   (DEVSPp only)
1203            * 2) or : <address>                           (DEVSPp only)
1204            * 3) or : <device word> [ (<seq num>) ]       (DEVSPp only)
1205            * 4) or : % <device type> [ (<seq num>) ]     (DEVSPp only)
1206            * 5) or : <assign word>                       (DEVSPp only)
1207            * 6) or : <device ID> [ (<seq num>) ]         (DEVSPp only)
1208            * 7) or [ : ] *                               (DEVSPp only)
1209            * 2) or <address>
1210            * 3) or <device word> [ (<seq num>) ]
1211            * 4) or % <device type> [ (<seq num>) ]
1212            * 5) or <assign word>
1213            * 6) or <device ID> [ (<seq num>) ]
1214            *
1215            *   Token output:
1216            * 1)    <string expression>
1217            * 2) or <tCOLON> <num expr>
1218            * 3) or <tCOLON> <tLITRL> <device word> [ <tCOLON> <num expr> ]
1219            * 4) or <tCOLON> <t%> <num expr> [ <tCOLON> <num expr> ]
1220            * 5) or <tCOLON> <tLITRL> <assign word>
1221            * 6) or <tCOLON> <tLITRL> <device ID> [ <tCOLON> <num expr> ]
1222            * 7) or <tCOLON> <t*>
1223            *
1224            ******************************
1225            *
1226            * Check for string expression first (Save state for restore)
1227            *
1228 F7857 7922 =FILSp  GOSUB  CKSTR        Check if string (Carry = NO)
1229 F785B 460         GOC    FILSPp       Not string...try literal
1230 F785E 6541        GOTO   FILSp8
1231            *-
1232            *-
1233 F7862      =FILSPp
1234 F7862 20          P=     0
1235 F7864 3100        LC(2)  =tLITRL      Literal token (File specifier)
1236 F7868 7F42        GOSUB  OUTBYT       Output it!
1237            *
1238            * Now D1 points to the first char of the file spec (or blanks)
1239            *
1240 F786C 2F          P=     15
1241 F786E 30A         LC(1)  10           10 characters max!
1242 F7871 74B1        GOSUB  NAMEpb       Parse the name (If carry, error)
1243            *
1244            * If carry is set, A[B] is the next char: could be bad first
1245            * char (digit) OR too long. I can't do either one...RTNSXM!
1246            *
1247 F7875 453         GOC    FILSpn       Not anything I understand
1248            *
1249            * Have parsed the name...check next character
```

```
1250                *
1251 F7878 104         R4=A                Save A[S] in R4[S]
1252 F787B 31A3        LCASC  \:\
1253 F787F 962         ?A=C   B            Is it a colon?
1254 F7882 D2          GOYES  FILSp0       Yes...continue
1255 F7884 31E2        LCASC  \.\
1256 F7888 966         ?A#C   B            Is it a "."?
1257 F788B 02          GOYES  FILSpn       No...return, set XM, clear carry
1258                *
1259                * Have a volume label...same rules as NAMES (alpha, alpha-digit)
1260                *
1261 F788D 171   =DVLBp  D1=D1+ 2            Skip the "."
1262                ****
1263                *
1264                *       LC(4)  (=tSEMIC)~(=tCOLON)
1265 F7890 33              NIBHEX 33
1266 F7892 00              CON(2) =tCOLON
1267 F7894 00              CON(2) =tSEMIC
1268                *
1269                ****
1270 F7896 7B22            GOSUB  oUT2TC
1271 F789A 2F              P=     15
1272 F789C 306             LC(1)  6            Max of 6 characters in volume lbl
1273 F789F 7A81            GOSUB  NAMEp
1274 F78A3 470             GOC    FILSpn       Bad first char OR too long..exit
1275                *
1276                * Check that at LEAST one char accepted
1277                *
1278 F78A6 94C             ?A#0   S            Any characters accepted?
1279 F78A9 F6              GOYES  FILSp!       Yes...check for loop #
1280                *
1281                * If here, had either a first char that was not a letter or
1282                * a colon OR had a name too long...either one is not HPIL.
1283                *
1284 F78AB 62F0 FILSpn   GOTO   FILSpX       Return, set XM, clear carry
1285                *-
1286                *-
1287 F78AF 171   FILSp0  D1=D1+ 2            Skip the colon
1288                *
1289                * Entry for Device parse (AFTER the colon)
1290                *
1291 F78B2 84A   =DEVSPp ST=0   =StarOK       FILE:* is NOT OK for this entry
1292 F78B5 7B03 =DVSPp  GOSUB  OUT:          Output the colon token
1293 F78B9 7382        GOSUB  Ntoken        Get next token...
1294 F78BD 3100        LC(2)  =t*
1295 F78C1 966         ?A#C   B            Is this a "*"?
1296 F78C4 71          GOYES  FILSp1       No...continue checking
1297                *
1298                * Found a "*"...is it permitted here?
1299                *
1300 F78C6 20              P=     =eILPAr      Illegal parameter
1301 F78C8 86A             ?ST=0  =StarOK
1302 F78CB C0              GOYES  FILSpx       Error if StarOK=0
1303                *
1304                * OK...output the token
```

```
1305             *
1306 F78CD 20           P=     0
1307 F78CF 78E1         GOSUB  OUTBYT        Output the t* token
1308 F78D3 64D0         GOTO   FILSp9        Done...exit
1309             *-
1310             *-
1311 F78D7 66C0 FILSpx  GOTO   FILSpX
1312             *-
1313             *-
1314 F78DB       FILSp1
1315             *
1316             * Not "*"...check if device type ("%")
1317             *
1318 F78DB 3100          LC(2)  =t%
1319 F78DF 966           ?A#C   B             Is it device type?
1320 F78E2 A5            GOYES  FILSp4        No...continue checking
1321             *
1322             * Device type (Syntax  %<num expr> [ (<num expr>) ] )
1323             *
1324 F78E4 76D1          GOSUB  oUT1TK        Output one token (t%)
1325             *
1326             * Following two lines are for stack levels (ENTERp,...)
1327             *
1328 F78E8 7B61          GOSUB  CKNUM+        Save info, call EXPPAR
1329 F78EC 7B71          GOSUB  CKNUM-        Check results of EXPPAR
1330 F78F0 46E           GOC    FILSpx        Error if carry (string/no expr)
1331 F78F3 3182 FILSp2  LCASC  \(\
1332 F78F7 966           ?A#C   B             Is there a sequence #?
1333 F78FA 22            GOYES  FILSp3        No...check for loop #
1334             *
1335             * Sequence # found
1336             *
1337 F78FC 74C2          GOSUB  OUT:          Output the "(" (kludge)
1338 F7900 7351          GOSUB  CKNUM+        Call EXPPAR (for stack levels)
1339 F7904 7361          GOSUB  CKNUM-        Check numeric expression
1340 F7908 4EC           GOC    FILSpx        Error if carry
1341             *
1342             * Check for closing paren now
1343             *
1344 F790B 3192          LCASC  \)\
1345 F790F 20            P=     =eMSPAr       Missing parameter
1346 F7911 966           ?A#C   B
1347 F7914 3C            GOYES  FILSpx        Error...no closing ")"
1348 F7916 20            P=     0
1349 F7918 7422 FILSp!  GOSUB  Ntoken        Get next token first
1350             *
1351             * Now check for loop #
1352             *
1353 F791C 31A3 FILSp3  LC(2)  \:\
1354 F7920 966           ?A#C   B             Is there a loop #?
1355 F7923 51            GOYES  FILsp8        No...exit after restoring D1
1356             *
1357             * Loop # found
1358             *
1359 F7925 3100          LC(2)  =tSEMIC       Internal representation
```

```
1360 F7929 7E81          GOSUB  OUTBYT        Output the semicolon token...
1361 F792D 7621          GOSUB  CKNUM+        Call EXPPAR (for stack levels)
1362 F7931 7631          GOSUB  CKNUM-        Check numeric expression
1363 F7935 486           GOC    FILSpX        Error if carry
1364 F7938 6B60 FILsp8   GOTO   FILSp8        Exit after restore
1365             *-
1366             *-
1367 F793C       FILSp4
1368             *
1369             * Not a device type...check further (Device word or address)
1370             *
1371             * First try address (if parses, then check for chars following)
1372             *
1373 F793C 7BD1          GOSUB  RESPTR        Restore pointer back to start
1374 F7940 7F41          GOSUB  SVDOD1        Save D0, D1
1375 F7944 7F01          GOSUB  CKNUM+        Call EXPPAR (for stack levels)
1376 F7948 7F11          GOSUB  CKNUM-        Check if numeric expression
1377 F794C 4A2           GOC    FILSp6        Not numeric...try device word
1378             *
1379             * Iff it is clearly a value expression (1,A+2,etc), then XM=1
1380             * (This means that any device ID's which begin with a numeric
1381             * function may need to be quoted)
1382             *
1383 F794F 831           ?XM=0
1384 F7952 50            GOYES  FILSp5        Not value expression...check more
1385 F7954 57C           GONC   FILSp3        Go always...this is an address
1386             *-
1387             *-
1388             *
1389             * If the next token is in [A-Z][0-9] and the previous char is
1390             * not a blank, then this must be a device ID
1391             *
1392 F7957 70C1 FILSp5   GOSUB  RESPTR        Back up to last token start
1393 F795B 14B           A=DAT1 B             Read the ASCII of the token
1394 F795E 72B1          GOSUB  cATCH+        Check if letter or digit next
1395 F7962 55B           GONC   FILSp!        No...this is address (check loop)
1396 F7965 1C1           D1=D1- 2
1397 F7968 14B           A=DAT1 B
1398 F796B 171           D1=D1+ 2
1399 F796E 3102          LC(2)  \ \           Check for a preceding blank
1400 F7972 962           ?A=C   B
1401 F7975 3A            GOYES  FILSp!        Blank...this is an address
1402             *
1403             * This is not an address...check if this is device word
1404             *
1405 F7977 7F21 FILSp6   GOSUB  RSDOD1        Restore D0, D1
1406 F797B 20            P=     0
1407 F797D 3100          LC(2)  =tLITRL
1408 F7981 7631          GOSUB  OUTBYT        Output the literal token first
1409 F7985 2F            P=     15
1410 F7987 308           LC(1)  8             Max of eight chars in device word
1411 F798A 7F90          GOSUB  NAMEp         Parse it
1412 F798E 4F0           GOC    FILSpX        Excess characters...error
1413             *
1414             * Check that at LEAST one character accepted
```

```
1415                *
1416 F7991 948         ?A=0   S            Any valid characters?
1417 F7994 A0          GOYES  FILSpX       No valid characters...error
1418 F7996 76A1        GOSUB  Ntoken       Get next token
1419 F799A 685F        GOTO   FILSp2       OK...check if sequence #
1420                *-
1421                *-
1422 F799E 21   FILSpX  P=     1
1423 F79A0 0D          P=P-1               Clear carry
1424 F79A2 00          RTNSXM
1425                *-
1426                *-
1427 F79A4 7371 FILSp8  GOSUB  RESPTR       Restore pointer
1428 F79A8 821  FILSp9  XM=0               Clear XM...
1429 F79AB 114         A=R4                Restore A[S] from R4[S]
1430                *
1431                * Entry for XWORD parse
1432                *
1433 F79AE      =XWORDp
1434 F79AE 03   =RTNCC  RTNCC              Return with carry clear
1435               ****************************************************************
1436               ****************************************************************
1437               **
1438               ** Name:       DVCSPp - Parse a device specifier (: optional)
1439               **
1440               ** Category:   STPARS
1441               **
1442               ** Purpose:
1443               **      Device spec parse...string expr, *, and [:] OK
1444               **
1445               ** Entry:
1446               **      D1 points to the ASCII character string
1447               **      D0 points to the location where the tokens go
1448               **      D[A] is the end of available memory
1449               **      P=0
1450               **
1451               ** Exit:
1452               **      D0 positioned past last token output by this routine
1453               **      D1 positioned past last character accepted
1454               **      Carry clear
1455               **      P=0
1456               **      Exits through ERRORP if error
1457               **
1458               ** Calls:      EOLCK,RESPTR,OUTBYT,CKSTR,BLANK,DVSPp,DVLBp
1459               **
1460               ** Uses.......
1461               **  Inclusive: A,B,C,D[15:5],R0-R3,D0,D1,P,ST[11,10,8,7,3:0],
1462               **             FUNCD0,PRMCNT[0]
1463               **
1464               ** Stk lvls:   5 (CKSTR)(DVSPp)
1465               **
1466               ** History:
1467               **
1468               **    Date     Programmer              Modification
1469               **  --------   ----------    ---------------------------------
```

```
1470            **  11/28/83      NZ          Updated documentation
1471            **
1472            ****************************************************************
1473            ****************************************************************
1474            *
1475            * Syntax:
1476            *   Input stream: <string expression>  or
1477            *       [ : ] <device specifier>  or
1478            *       [ : ] {*}  or
1479            *       . <volume label>
1480            *   Token output: <string expression>  or
1481            *       <tCOLON> <device specifier>  or
1482            *       <tCOLON> <t*>  or
1483            *       <tCOLON> <tSEMIC> <volume label>
1484            *
1485 F79B0      =PACKp
1486 F79B0 84A  =DVCPn* ST=0   =StarOK
1487 F79B3 6600         GOTO   DVCSPp
1488            *-
1489            *-
1490 F79B7 85A  =DVCPy* ST=1   =StarOK       "*" OK
1491 F79BA 848  =DVCSPp ST=0   =OptDev       Device specifier required
1492            *
1493 F79BD 8F00 DVCSPc  GOSBVL =EOLCK        Check if is EOL, @, !, ELSE
         000
1494 F79C4 5B1          GONC   DVCP05       If not, restore ptr and cont.
1495 F79C7 878          ?ST=1  =OptDev      Is device spec. optional ?
1496 F79CA 60           GOYES  DVCSPr       If so, we are done
1497 F79CC 6DCB         GOTO   MSGPAR       Otherwise say, Missing Parm.
1498            *-
1499            *-
1500 F79D0 7741 DVCSPr  GOSUB  RESPTR       Restore  pointer for device parse
1501 F79D4 20           P=     0            Load dummy comma token into C
1502 F79D6 3100         LC(2)  =tCOMMA
1503 F79DA 7DD0         GOSUB  OUTBYT       Output the comma token
1504 F79DE 03           RTNCC               Already restored input pointer
1505            *-
1506            *-
1507 F79E0 7731 DVCP05  GOSUB  RESPTR       Restore pointer
1508 F79E4 7C90         GOSUB  CKSTR        Check if string (Carry=NO)
1509 F79E8 460          GOC    DVCP10       No...try literal
1510 F79EB 6F21         GOTO   RESPTR       Yes...restore pointer, return
1511            *-
1512            *-
1513 F79EF 7731 DVCP10  GOSUB  BLANK        Read in the character
1514 F79F3 31E2         LCASC  \.\          Check first for volume label
1515 F79F7 962          ?A=C   B            Is this a volume label?
1516 F79FA 22           GOYES  DVCP40       Yes...volume label
1517 F79FC 31A3         LCASC  \:\
1518 F7A00 966          ?A#C   B            Is there a colon?
1519 F7A03 50           GOYES  DVCP30       No...continue
1520            *
1521            * Colon is present...skip it
1522            *
1523 F7A05 171          D1=D1+ 2            Skip to next item
```

```
1524 F7A08 79AE DVCP30  GOSUB  DVSPp         Device spec parse
1525 F7A0C 4B0  DVCP35  GOC    DVCP65        If carry, error (can't happen)
1526 F7A0F 831          ?XM=0                OK?  Processed as is?
1527 F7A12 21           GOYES  DVCP70        Yes...return with carry clear
1528 F7A14 628B         GOTO   INITp1        If not, say "Syntax"
1529            *-
1530            *-
1531 F7A18 667B DVCP65  GOTO   Error!        Parse error, already set up
1532            *-
1533            *-
1534 F7A1C      DVCP40
1535            *
1536            * Volume label
1537            *
1538 F7A1C 7D6E         GOSUB  DVLBp         Device volume label parse
1539 F7A20 6BEF         GOTO   DVCP35        Go check for error
1540            *-
1541            *-
1542 F7A24 84A  DVCP70  ST=0   10            ST(10) MUST be zero (Implied LET)
1543 F7A27 03           RTNCC
1544            ********************************************************************
1545            ********************************************************************
1546            **
1547            ** Name:       NAMEpb - Skip leading blanks, parse device word
1548            ** Name:       NAMEp - Parse a device word (C[S] is # chars)
1549            **
1550            ** Category:   PARUTL
1551            **
1552            ** Purpose:
1553            **      Parse a device word: <letter > {<letter> | <digit >} *n
1554            **
1555            ** Entry:
1556            **      C[S] is max number of characters to accept
1557            **      D1 points to the ASCII character string
1558            **      D0 points to the location where the tokens go
1559            **      D[A] is the end of available memory
1560            **
1561            ** Exit:
1562            **      First character not used in A[B] (char @ D1)
1563            **      Carry set if length exceeded or first char is a digit
1564            **      A[S]=0 if no chars, #F if characters
1565            **      D0 positioned past last character output by this routine
1566            **      D1 positioned past last character accepted
1567            **      P=0
1568            **
1569            ** Calls:      BLANK,CATC++,OUT1TK
1570            **
1571            ** Uses.......
1572            **  Inclusive: A[S,B],C[S,B],P,D0,D1,ST[2:1]
1573            **
1574            ** Stk lvls:   3 (CATC++)
1575            **
1576            ** History:
1577            **
1578            **    Date     Programmer            Modification
```

```
1579                ** --------   ----------   --------------------------------
1580                ** 11/28/83      NZ         Updated documentation
1581                **
1582                ********************************************************************
1583                ********************************************************************
1584                *
1585                * Syntax:
1586                *   Input stream: [ <letter> [ <letter> | <digit> ] *n ]
1587                *   Token output: Same as input (with all letters converted to
1588                *       upper case)
1589                *
1590 F7A29 7DF0 =NAMEpb GOSUB  BLANK          Skip any leading blanks!
1591 F7A2D 20   =NAMEp  P=     0
1592 F7A2F AC0          A=0    S            Clear "char" flag
1593 F7A32 7BD0         GOSUB  CATC++       Read first char, set statuses
1594 F7A36 500          RTNNC               Not letter or digit...return, CC
1595 F7A39 871          ?ST=1  Digit        Is this a digit?
1596 F7A3C 00           RTNYES              Yes...not permitted here-Set Carry
1597 F7A3E A4C          A=A-1  S            Set A[S]="F"
1598 F7A41 A4E  NAMEp1  C=C-1  S            Decrement count
1599 F7A44 400          RTNC                Error...too long! (Set Carry)
1600 F7A47 7370         GOSUB  OUT1TK       Output the token
1601 F7A4B 171          D1=D1+ 2            Increment to next token
1602 F7A4E 7FB0         GOSUB  CATC++       Read it, check it out
1603 F7A52 4EE          GOC    NAMEp1       Letter or digit...OK!
1604 F7A55 03           RTNCC               Carry clear = OK!
1605                ********************************************************************
1606                ********************************************************************
1607                **
1608                ** Name:       CKNUM - Check for a numeric expr (output it)
1609                ** Name:       CKNUM+ - Save D1 in R3, goto EXPPAR
1610                ** Name:       CKNUM- - Check EXPPAR exit conditions for number
1611                **
1612                ** Category:   LOCAL
1613                **
1614                ** Purpose:
1615                **      Check for a numeric expression and output the tokens
1616                **      for that expression
1617                **
1618                ** Entry:
1619                **      D1 points to the ASCII character string
1620                **      D0 points to the location where the tokens go
1621                **      D[A] is the end of available memory
1622                **      P=0
1623                **
1624                ** Exit:
1625                **      Carry set if not numeric (P is error number for parse,
1626                **        D1 points to the error)
1627                **      Carry clear if OK (tokens output, D0,D1 set to next
1628                **        items, P=0)
1629                **      D0 positioned past last token output by this routine
1630                **      D1 positioned past last character accepted
1631                **
1632                ** Calls:      EXPPAR
1633                **
```

```
1634            ** Uses.......
1635            **  Inclusive: A,B,C,D[15:5],R0,R1,R3,D0,D1,P,ST[11,7,3:0],
1636            **             FUNCD0,PRMCNT[0]
1637            **
1638            ** Stk lvls:   CKNUM:  4 (CKNUM+)
1639            ** Stk lvls:   CKNUM+: 3 (<EXPPAR>)
1640            ** Stk lvls:   CKNUM-: 0
1641            **
1642            ** History:
1643            **
1644            **    Date     Programmer            Modification
1645            **  --------   ----------   ---------------------------------
1646            **  11/28/83      NZ        Updated documentation
1647            **
1648            ****************************************************************
1649            ****************************************************************
1650 F7A57 137  CKNUM+  CD1EX
1651 F7A5A 10B          R3=C                Save input pointer for case of
1652 F7A5D 135          D1=C                  string (to set error pointer)
1653 F7A60 8D00 Exppar  GOVLNG =EXPPAR
          000
1654            *-
1655            *-
1656 F7A67 7CEF CKNUM   GOSUB  CKNUM+        Call EXPPAR after save
1657            *
1658 F7A6B 873  CKNUM-  ?ST=1  NumExp        Is it numeric?
1659 F7A6E B0           GOYES  CKNUM1        Yes...OK
1660 F7A70 11B          C=R3
1661 F7A73 135          D1=C                Restore input pointer
1662 F7A76 590          GONC   CKNUM2        Go always
1663            *-
1664            *-
1665 F7A79 870  CKNUM1  ?ST=1  InvalE        Invalid?
1666 F7A7C 40           GOYES  CKNUM2        Yes...error
1667 F7A7E 03           RTNCC                No...all OK
1668            *-
1669            *-
1670 F7A80 20   CKNUM2  P=     =eILEXp       Illegal expression
1671 F7A82 02           RTNSC
1672            ****************************************************************
1673            ****************************************************************
1674            **
1675            ** Name:       CKSTR - Parse a string expression
1676            **
1677            ** Category:   LOCAL
1678            **
1679            ** Purpose:
1680            **      CKSTR tries to parse a string expression non-destructivel
1681            **
1682            ** Entry:
1683            **      D1 points to the ASCII character string
1684            **      D0 points to the location where the tokens go
1685            **      D[A] is the end of available memory
1686            **      P=0
1687            **
```

```
1688            ** Exit:
1689            **      Carry set if not string (D0, D1 restored)
1690            **      Carry clear if string (tokens output)
1691            **      D0 positioned past last token output by this routine
1692            **      D1 positioned past last character accepted
1693            **      P=0
1694            **      Exits through ERRORP if error
1695            **
1696            ** Calls:      SVDOD1,EXPPAR,<RSDOD1>
1697            **
1698            ** Uses.......
1699            **  Inclusive: A,B,C,D[15:5],R0-R2,D0,D1,P,ST[11,7,3:0],FUNCD0,
1700            **             PRMCNT[0]
1701            **
1702            ** Stk lvls:   4 (EXPPAR)
1703            **
1704            ** History:
1705            **
1706            **    Date     Programmer              Modification
1707            **  --------   ----------    --------------------------------
1708            **  11/28/83      NZ         Updated documentation
1709            **
1710            ****************************************************************
1711            ****************************************************************
1712 F7A84 7B00 =CKSTR  GOSUB  SVDOD1        Save D0 and D1 in R2
1713 F7A88 74DF         GOSUB  Exppar
1714 F7A8C 873          ?ST=1  NumExp        Valid numeric? (set unless string)
1715 F7A8F B1           GOYES  RSDOD1        Yes...not string
1716 F7A91 03           RTNCC                Return (valid string)
1717            ****************************************************************
1718            ****************************************************************
1719            **
1720            ** Name:       SVDOD1 - Save D0 and D1 in R2
1721            ** Name:       RSDOD1 - Restore D0 and D1 from R2
1722            **
1723            ** Category:   STPARS
1724            **
1725            ** Purpose:
1726            **      Save/restore D0 and D1 in/from R2
1727            **
1728            ** Entry:
1729            **      SVDOD1: none
1730            **      RSDOD1: R2 contains D0 and D1 (from SVDOD1)
1731            **
1732            ** Exit:
1733            **      SVDOD1: R2 contains D0 and D1 values
1734            **      RSDOD1: D0 and D1 are restored from R2
1735            **      P,Carry unchanged from input
1736            **
1737            ** Calls:      CSLC5,CSRC5
1738            **
1739            ** Uses.......
1740            **  Inclusive: C[W],R2
1741            **
1742            ** Stk lvls:   1 (CSLC5)(CSRC5)
```

```
1743                    **
1744                    ** History:
1745                    **
1746                    **    Date     Programmer            Modification
1747                    **  --------   ----------   ---------------------------------
1748                    **  11/28/83      NZ        Added documentation
1749                    **
1750                    ****************************************************************
1751                    ****************************************************************
1752 F7A93 137       =SVDOD1 CD1EX
1753 F7A96 135               D1=C
1754 F7A99 8E00              GOSUBL =CSLC5          Save D1 in R2[9:5]
          00
1755 F7A9F 136               CD0EX
1756 F7AA2 134               D0=C                   Save D0 in R2[A]
1757 F7AA5 10A               R2=C
1758 F7AA8 01                RTN
1759                    *-
1760                    *-
1761 F7AAA 11A       =RSD0D1 C=R2
1762 F7AAD 134               D0=C                   Restore D0
1763 F7AB0 8E00              GOSUBL =CSRC5
          00
1764 F7AB6 135               D1=C                   Restore D1
1765 F7AB9 01                RTN
1766                    ****************************************************************
1767                    *
1768                    * These routines are identical to the mainframe routines by the
1769                    * same names
1770                    *
1771                    ****************************************************************
1772 F7ABB AEE         =OUTBYT ACEX   B
1773 F7ABE 8D00        =oUT1TK GOVLNG =OUT1TK
          000
1774                    *-
1775                    *-
1776 F7AC5 8D00        =oUT2TC GOVLNG =OUT2TC
          000
1777                    *-
1778                    *-
1779 F7ACC AFA         =OUT3TC A=C    W
1780 F7ACF 8D00        =oUT3TK GOVLNG =OUT3TK
          000
1781                    *-
1782                    *-
1783 F7AD6 AFA         =OUTNBC A=C    W
1784 F7AD9 8D00        =oUTNBS GOVLNG =OUTNBS
          000
1785                    *-
1786                    *-
1787                    ****************************************************************
1788                    ****************************************************************
1789                    **
1790                    ** Name:       NUMCK+ - Restore input pointer, check num expr
1791                    ** Name:       NUMCK - Check for a valid numeric expression
```

```
1792                    **
1793                    ** Purpose:
1794                    **      Check for a valid numeric expression. If not found,
1795                    **      then exit to ERRORR
1796                    **
1797                    ** Entry:
1798                    **      D1 points to the ASCII character string
1799                    **      D0 points to the location where the tokens go
1800                    **      D[A] is the end of available memory
1801                    **      P=0
1802                    **
1803                    ** Exit:
1804                    **      D0 positioned past last token output by this routine
1805                    **      D1 positioned past last character accepted
1806                    **      P=0
1807                    **      Carry clear
1808                    **      Exits through ERRORR if error
1809                    **
1810                    ** Calls:       RESPTR,EXPPAR
1811                    **
1812                    ** Uses.......
1813                    **  Inclusive: A,B,C,D[15:5],R0,R1,R3,D0,D1,P,ST[11,7,3:0],
1814                    **             FUNCD0,PRMCNT[0]
1815                    **
1816                    ****************************************************************
1817                    ****************************************************************
1818 F7AE0 7730  =NUMCK+ GOSUB  RESPTR
1819 F7AE4 11B   =NUMCK  C=R3                  Preserve upper part of R3
1820 F7AE7 137          CD1EX
1821 F7AEA 135          D1=C                  Save for case of string expression
1822 F7AED 10B          R3=C
1823 F7AF0 7C6F         GOSUB  Exppar         Mainframe jump to EXPPAR
1824 F7AF4 873          ?ST=1  NumExp         Numeric?
1825 F7AF7 B0           GOYES  NUMCK1         Yes...check if valid
1826 F7AF9 11B          C=R3                  No...restore D1 (string expr)
1827 F7AFC 135          D1=C
1828 F7AFF 590          GONC   NUMCK2         Go always
1829                    *-
1830                    *-
1831 F7B02 870  NUMCK1  ?ST=1  InvalE         Invalid expression?
1832 F7B05 40           GOYES  NUMCK2         Yes...error
1833 F7B07 03           RTNCC                 No...valid numeric expression
1834                    *-
1835                    *-
1836 F7B09 20   NUMCK2  P=     =eILEXp        Illegal expression
1837 F7B0B 8C00         GOLONG =ERRORR        Don't restore D1 (already set)
           00
1838                    *
1839                    * More duplicates of mainframe routines
1840                    *
1841 F7B11 14B  =CATC++ A=DAT1 B
1842 F7B14 8D00 =cATCH+ GOVLNG =CATCH+
           000
1843                    ****************************************************************
1844                    ****************************************************************
```

```
1845                ** 
1846                ** Name:       RESPTR - Restore D1 from LEXPTR
1847                ** 
1848                ** Category:   LOCAL
1849                ** 
1850                ** Purpose:
1851                **      Restore the input pointer from LEXPTR
1852                ** 
1853                ** Entry:
1854                **      None
1855                ** 
1856                ** Exit:
1857                **      D1 restored from LEXPTR
1858                **      Carry clear
1859                ** 
1860                ** Calls:      None
1861                ** 
1862                ** Uses.......
1863                **  Inclusive: A[A],D1
1864                ** 
1865                ** Stk lvls:   0
1866                ** 
1867                ** History:
1868                ** 
1869                **    Date     Programmer              Modification
1870                **  --------   ----------    ---------------------------------
1871                **  11/28/83      NZ         Added documentation
1872                ** 
1873                ******************************************************************
1874                ******************************************************************
1875 F7B1B 1F00 =RESPTR D1=(5) =LEXPTR
          000
1876 F7B22 143          A=DAT1 A
1877 F7B25 131          D1=A
1878 F7B28 03           RTNCC
1879                ******************************************************************
1880                ******************************************************************
1881                ** 
1882                ** Name:       BLANK - Skip blanks, return first non-blank char
1883                ** 
1884                ** Category:   PARUTL
1885                ** 
1886                ** Purpose:
1887                **      Skip blanks in the input stream
1888                ** 
1889                ** Entry:
1890                **      D1 points to the input stream
1891                ** 
1892                ** Exit:
1893                **      A[B] contains the next character
1894                **      D1 points to the character in A[B]
1895                ** 
1896                ** Calls:      None
1897                ** 
1898                ** Uses.......
```

```
1899            **  Inclusive: A[B],C[B],P,D1 (D1 only if leading blanks)
1900            **
1901            ** Stk lvls:   0
1902            **
1903            ** History:
1904            **
1905            **    Date     Programmer              Modification
1906            **  --------   ----------    --------------------------------
1907            **  11/28/83      NZ         Updated documentation
1908            **
1909            ****************************************************************
1910            ****************************************************************
1911 F7B2A 20   =BLANK  P=     0
1912 F7B2C 3102         LCASC  \ \
1913 F7B30 1C1          D1=D1- 2
1914 F7B33 171  Skip    D1=D1+ 2
1915 F7B36 14B  =SKIP   A=DAT1 B
1916 F7B39 962          ?A=C   B
1917 F7B3C 7F           GOYES  Skip
1918 F7B3E 01           RTN
1919            *-
1920            *-
1921 F7B40 8D00 Ntoken  GOVLNG =NTOKEN
           000
1922            ****************************************************************
1923            ****************************************************************
1924            **
1925            ** Name:       ENABLp - Parse the ENABLE INTR statement
1926            **
1927            ** Category:   STPARS
1928            **
1929            ** Purpose:
1930            **      Parse the ENABLE INTR statement
1931            **
1932            ** Entry:
1933            **      D1 points to the ASCII character string
1934            **      D0 points to the location where the tokens go
1935            **      D[A] is the end of available memory
1936            **      P=0
1937            **
1938            ** Exit:
1939            **      D0 positioned past last token output by this routine
1940            **      D1 positioned past last character accepted
1941            **      P=0
1942            **      Exits through ERRORP if error
1943            **
1944            ** Calls:      WRDSCN,<REQSTp>
1945            **
1946            ** Uses.......
1947            **  Inclusive: A,B,C,D[15:5],R0,R1,R2,D0,D1,P,ST[11,7,3:0],
1948            **             FUNCD0,PRMCNT[0]
1949            **
1950            ** Stk lvls:   5 (<REQSTp>)
1951            **
1952            ** History:
```

```
1953                **
1954                **    Date     Programmer             Modification
1955                **  --------   ----------   ------------------------------
1956                **  11/28/83      NZ        Added documentation
1957                **
1958                ********************************************************************
1959                ********************************************************************
1960 F7B47 7840 =ENABLp GOSUB  wrdscn
1961 F7B48 00           CON(2) =tXWORD
1962 F7B4D 00           CON(2) =LEXPIL
1963 F7B4F 00           CON(2) =tINTRR
1964 F7B51 900          REL(3) ENBLp1
1965 F7B54 00           CON(2) 00
1966 F7B56 607A         GOTO   INITp1        Syntax error
1967                *-
1968                *-
1969 F7B5A 185  ENBLp1  D0=D0- 6             Don't output the INTR token
1970                *
1971                * Fall into REQUEST parse (ENABLE and REQUEST match after INTR)
1972                *
1973                ********************************************************************
1974                ********************************************************************
1975                **
1976                ** Name:       REQSTp - Parse the REQUEST statement
1977                **
1978                ** Category:   STPARS
1979                **
1980                ** Purpose:
1981                **      Parse the REQUEST statement
1982                **
1983                ** Entry:
1984                **      D1 points to the ASCII character string
1985                **      D0 points to the location where the tokens go
1986                **      D[A] is the end of available memory
1987                **      P=0
1988                **
1989                ** Exit:
1990                **      D0 positioned past last token output by this routine
1991                **      D1 positioned past last character accepted
1992                **      P=0
1993                **      Exits through ERRORP if error
1994                **
1995                ** Calls:      LOOP#p,ST!NOp,<RESPTR>
1996                **
1997                ** Uses.......
1998                **  Inclusive: A,B,C,D[15:5],R0,R1,R2,D0,D1,P,ST[11,7,3:0],
1999                **             FUNCD0,PRMCNT[0]
2000                **
2001                ** Stk lvls:   6 (LOOP#p)
2002                **
2003                ** History:
2004                **
2005                **    Date     Programmer             Modification
2006                **  --------   ----------   ------------------------------
2007                **  11/28/83      NZ        Added documentation
```

```
2008                **
2009                ************************************************************
2010                ************************************************************
2011                *
2012                * ENABLE parse falls into REQUEST parse
2013                *
2014 F7B5D 7BDB =REQSTp GOSUB  LOOP#p
2015 F7B61 84A          ST=0   =StrOK
2016 F7B64 74CC         GOSUB  ST!NOp        Check for a string or number
2017 F7B68 460          GOC    REQp10        Error if carry
2018 F7B6B 6FAF         GOTO   RESPTR        Restore pointer if OK
2019                *-
2020                *-
2021 F7B6F 699F REQp10  GOTO   NUMCK2
2022                ************************************************************
2023                ************************************************************
2024                **
2025                ** Name:       PASSp - Parse the PASS CONTROL statement
2026                **
2027                ** Category:   STPARS
2028                **
2029                ** Purpose:
2030                **      Parse the PASS CONTROL statement
2031                **
2032                ** Entry:
2033                **      D1 points to the ASCII character string
2034                **      D0 points to the location where the tokens go
2035                **      D[A] is the end of available memory
2036                **      P=0
2037                **
2038                ** Exit:
2039                **      D0 positioned past last token output by this routine
2040                **      D1 positioned past last character accepted
2041                **      P=0
2042                **      Exits through ERRORP if error
2043                **
2044                ** Calls:      WRDSCN,<DVCSPc>
2045                **
2046                ** Uses.......
2047                **  Inclusive: A,B,C,D[15:5],R0-R4,D0,D1,P,ST[11:7,3:0],
2048                **             FUNCD0,PRMCNT[0]
2049                **
2050                ** Stk lvls:   5 (<DVCSPc>)
2051                **
2052                ** History:
2053                **
2054                **    Date     Programmer             Modification
2055                **  --------   ----------   ---------------------------------
2056                **  11/28/83      NZ        Added documentation
2057                **
2058                ************************************************************
2059                ************************************************************
2060 F7B73 7C10 =PASSp  GOSUB  wrdscn
2061 F7B77 00          CON(2) =tXWORD
2062 F7B79 00          CON(2) =LEXPIL
```

```
2063 F7B7B 00          CON(2) =tCNTRL
2064 F7B7D 900         REL(3) PASp10        ADDRESS FOR MATCHING
2065 F7B80 00          CON(2) 00
2066 F7B82 671A PASpER GOTO   MSGPAR        Missing parameter
2067           *-
2068           *-
2069 F7B86 185  PASp10 D0=D0- 6             Don't need the tCNTRL
2070 F7B89 84A         ST=0   =StarOK       "*" is not OK here
2071 F7B8C 858         ST=1   =OptDev       Device spec is optional
2072 F7B8F 6D2E        GOTO   DVCSPc
2073           *-
2074           *-
2075 F7B93 8D00 wrdscn GOVLNG =WRDSCN
       000
2076           ****************************************************************
2077           ****************************************************************
2078           **
2079           ** Name:       CNTRLp - Parse the CONTROL ON/OFF statement
2080           ** Name:       RESTp - Parse the RESTORE IO statement
2081           **
2082           ** Category:   STPARS
2083           **
2084           ** Purpose:
2085           **      Parse the CONTROL ON/OFF or RESTORE IO statement
2086           **
2087           ** Entry:
2088           **      D1 points to the ASCII character string
2089           **      D0 points to the location where the tokens go
2090           **      D[A] is the end of available memory
2091           **      P=0
2092           **
2093           ** Exit:
2094           **      D0 positioned past last token output by this routine
2095           **      D1 positioned past last character accepted
2096           **      P=0
2097           **      If no error, carry clear
2098           **      Exits through ERRORP if error
2099           **
2100           ** Calls:      WRDSCN,EOLCK,NUMCK+,<RESPTR>
2101           **
2102           ** Uses.......
2103           **  Inclusive: A,B,C,D[15:5],R0-R2,R3[A],D0,D1,P,ST[11,7,3:0],
2104           **             FUNCD0,PRMCNT[0]
2105           **
2106           ** Stk lvls:   5 (NUMCK+)
2107           **
2108           ** History:
2109           **
2110           **    Date     Programmer            Modification
2111           **  --------   ----------    --------------------------------
2112           **  11/28/83      NZ         Added documentation
2113           **
2114           ****************************************************************
2115           ****************************************************************
2116 F7B9A 75FF =CNTRLp GOSUB  wrdscn
```

```
2117 F7B9E 00          CON(2) =tON
2118 F7BA0 210         REL(3) CNTROL       CONTROL ON
2119 F7BA3 00          CON(2) =tOFF
2120 F7BA5 D00         REL(3) CNTROL       CONTROL OFF
2121 F7BA8 00          CON(2) 00
2122 F7BAA 67DF        GOTO   PASpER       "Missing Parameter"
2123           *-
2124           *-
2125 F7BAE 79AA =RESTp GOSUB  IOp          First parse "IO"
2126           *
2127           * Check for optional numeric expression
2128           *
2129 F7BB2 8F00 CNTROL GOSBVL =EOLCK       See if reached end-of-statement
               000
2130 F7BB9 460         GOC    Resptr      Yes...done
2131 F7BBC 702F        GOSUB  NUMCK+      Must be a numeric expr
2132 F7BC0 6A5F Resptr GOTO   RESPTR
2133           *-
2134           *-
2135 F7BC4 3100 OUT:   LC(2)  =tCOLON
2136 F7BC8 62FE        GOTO   OUTBYT
2137           ****************************************************************
2138           ****************************************************************
2139           **
2140           ** Name:       CONWUC - Convert A[W] to upper case
2141           **
2142           ** Category:   PILUTL
2143           **
2144           ** Purpose:
2145           **      Convert A[W] to upper case
2146           **
2147           ** Entry:
2148           **      P=0
2149           **      D1 points at the letters and digits to convert
2150           **
2151           ** Exit:
2152           **      A[W] in upper case
2153           **      P=0
2154           **      Carry clear
2155           **
2156           ** Calls:      <CNVWUC>
2157           **
2158           ** Uses.......
2159           **  Inclusive: A[W],C[W]
2160           **
2161           ** Stk lvls:   1 <CNVWUC>
2162           **
2163           ** History:
2164           **
2165           **    Date     Programmer             Modification
2166           **  --------   ----------    ---------------------------------
2167           **  09/07/83      NZ         Changed entry to read data at D1
2168           **                           first, then convert to upper case
2169           **  09/06/83      NZ         Changed to goto mainframe routine
2170           **  01/03/83      NZ         Updated documentation
```

```
2171            **
2172            ************************************************************************
2173            ************************************************************************
2174 F7BCC 8D00 =CONWUC GOVLNG =CNVWUC       Convert to upper case (mainframe)
           000
2175 F7BD3              END
```

```
 ?A=CM+  Ext                  -   223   802
=ASGNp   Abs 1013404 #F769C  -   655
 ASGNp1  Abs 1013436 #F76BC  -   669   664
 ASGNp2  Abs 1013444 #F76C4  -   673   660
=BLANK   Abs 1014570 #F7B2A  -  1911   453   822   838   926  1053  1139  1513
                             1590
=CATC++  Abs 1014545 #F7B11  -  1841  1593  1602
 CATCH+  Ext                 -  1842
 CKNUM   Abs 1014375 #F7A67  -  1656   230   239   351   904
 CKNUM+  Abs 1014359 #F7A57  -  1650  1328  1338  1361  1375  1656
 CKNUM-  Abs 1014379 #F7A6B  -  1658  1329  1339  1362  1376
 CKNUM1  Abs 1014393 #F7A79  -  1665  1659
 CKNUM2  Abs 1014400 #F7A80  -  1670  1662  1666
=CKSTR   Abs 1014404 #F7A84  -  1712   659  1228  1508
=CLEARp  Abs 1013278 #F761E  -   409
=CNTRLp  Abs 1014682 #F7B9A  -  2116
 CNTROL  Abs 1014706 #F7BB2  -  2129  2118  2120
 CNVWUC  Ext                 -  2174
=CONWUC  Abs 1014732 #F7BCC  -  2174   201   454   828
 CSLC5   Ext                 -  1754
 CSRC5   Ext                 -  1763
=DEVSPp  Abs 1013938 #F78B2  -  1291
 DISPP   Ext                 -   118
 DVCP05  Abs 1014240 #F79E0  -  1507  1494
 DVCP10  Abs 1014255 #F79EF  -  1513  1509
 DVCP30  Abs 1014280 #F7A08  -  1524  1519
 DVCP35  Abs 1014284 #F7A0C  -  1525  1539
 DVCP40  Abs 1014300 #F7A1C  -  1534  1516
 DVCP65  Abs 1014296 #F7A18  -  1531  1525
 DVCP70  Abs 1014308 #F7A24  -  1542  1527
=DVCPn*  Abs 1014192 #F79B0  -  1486   116   124
=DVCPy*  Abs 1014199 #F79B7  -  1490    73
 DVCSPc  Abs 1014205 #F79BD  -  1493   414  2072
=DVCSPp  Abs 1014202 #F79BA  -  1491  1487
 DVCSPr  Abs 1014224 #F79D0  -  1500  1496
=DVLBp   Abs 1013901 #F788D  -  1261  1538
=DVSPp   Abs 1013941 #F78B5  -  1292  1524
=Digit   Abs       1 #00001  -    18  1595
=ENABLp  Abs 1014599 #F7B47  -  1960
 ENBLp1  Abs 1014618 #F7B5A  -  1969  1964
=ENTERp  Abs 1013021 #F751D  -   124
 ENTR10  Abs 1013042 #F7532  -   129   127
 EOLCK   Ext                 -  1493  2129
 ERROR!  Ext                 -   212
 ERRORP  Ext                 -   670
 ERRORR  Ext                 -  1837
 EXPPAR  Ext                 -  1653
=EolOK   Abs       9 #00009  -    29   823  1031  1048
 Error!  Abs 1013135 #F758F  -   212  1531
 Errorp  Abs 1013438 #F76BE  -   670   161   218   241   245
 Errorx  Abs 1013354 #F766A  -   561   458   610
 Exppar  Abs 1014368 #F7A60  -  1653  1117  1713  1823
=ExprOK  Abs       8 #00008  -    28   785  1032  1035
=FILSPp  Abs 1013858 #F7862  -  1233  1229
=FILSp   Abs 1013847 #F7857  -  1228   210
```

```
 FILSp!  Abs 1014040 #F7918 -  1349  1279  1395  1401
 FILSp0  Abs 1013935 #F78AF -  1287  1254
 FILSp1  Abs 1013979 #F78DB -  1314  1296
 FILSp2  Abs 1014003 #F78F3 -  1331  1419
 FILSp3  Abs 1014044 #F791C -  1353  1333  1385
 FILSp4  Abs 1014076 #F793C -  1367  1320
 FILSp5  Abs 1014103 #F7957 -  1392  1384
 FILSp6  Abs 1014135 #F7977 -  1405  1377
 FILSp8  Abs 1014180 #F79A4 -  1427  1230  1364
 FILSp9  Abs 1014184 #F79A8 -  1428  1308
 FILSpX  Abs 1014174 #F799E -  1422  1284  1311  1363  1412  1417
 FILSpn  Abs 1013931 #F78AB -  1284  1247  1257  1274
 FILSpx  Abs 1013975 #F78D7 -  1311  1302  1330  1340  1347
 FILsp8  Abs 1014072 #F7938 -  1364  1355
 FRAMEE  Ext                -  1012
 FRASP1  Abs 1013635 #F7783 -   987   993
 FRASP2  Abs 1013663 #F779F -   999  1002
 FRASP3  Abs 1013793 #F7821 -  1056  1013
 FRASPn  Abs 1013778 #F7812 -  1050  1047
=FRASPp  Abs 1013609 #F7769 -   975   777   842
 FRASPx  Abs 1013740 #F77EC -  1036  1034
 FRASPy  Abs 1013775 #F780F -  1049  1039
 INITP.  Abs 1013141 #F7595 -   215   211
 INITP0  Abs 1013152 #F75A0 -   221   216
 INITP2  Abs 1013156 #F75A4 -   222   240
 INITPE  Abs 1013187 #F75C3 -   241   231
=INITPR  Abs 1013176 #F75B8 -   232   224   353
=INITp   Abs 1013105 #F7571 -   201
 INITp1  Abs 1013191 #F75C7 -   244   205  1528  1966
=IOp     Abs 1013339 #F765B -   555   655  2125
 IOp10   Abs 1013363 #F7673 -   565   559
 IOp20   Abs 1013366 #F7676 -   566   505
=InvalE  Abs       0 #00000 -    17  1118  1665  1831
 LEXPIL  Ext                -   340   503   557   606  1962  2062
 LEXPTR  Ext                -  1875
=LOCALp  Abs 1013225 #F75E9 -   332
 LOCLp1  Abs 1013274 #F761A -   362   345
 LOOP#1  Abs 1013601 #F7761 -   921   905   913
 LOOP#2  Abs 1013605 #F7765 -   926   918
=LOOP#p  Abs 1013564 #F773C -   900   284   767  2014
 Loopp   Abs 1013249 #F7601 -   350   463
 MSGPAR  Abs 1013146 #F759A -   217  1497  2066
=NAMEp   Abs 1014317 #F7A2D -  1591  1273  1411
 NAMEp1  Abs 1014337 #F7A41 -  1598  1603
=NAMEpb  Abs 1014313 #F7A29 -  1590  1242
 NTOKEN  Ext                -  1921
=NUMCK   Abs 1014500 #F7AE4 -  1819
=NUMCK+  Abs 1014496 #F7AE0 -  1818  2131
 NUMCK1  Abs 1014530 #F7B02 -  1831  1825
 NUMCK2  Abs 1014537 #F7B09 -  1836  1828  1832  2021
 Ntoken  Abs 1014592 #F7B40 -  1921    69   143   221   333   614   661  1293
                            -  1349  1418
=NumExp  Abs       3 #00003 -    20  1120  1658  1714  1824
=OFFIOp  Abs 1013320 #F7648 -   501
=ONINTp  Abs 1013368 #F7678 -   604
```

```
 ONINp1  Abs 1013387 #F768B -   613   608
 ONP40   Ext                -   616
 OUT1TK  Ext                -  1773
 OUT2TC  Ext                -  1776
=OUT3TC  Abs 1014476 #F7ACC -  1779
 OUT3TK  Ext                -  1780
 OUT:    Abs 1014724 #F7BC4 -  2135   665   834   977  1292  1337
=OUTBYT  Abs 1014459 #F7ABB -  1772   130   156   359   666   903  1116  1236
                              1307  1360  1408  1503  2136
=OUTNBC  Abs 1014486 #F7AD6 -  1783
 OUTNBS  Ext                -  1784
=OUTPp   Abs 1013006 #F750E -   116
 OUTpCK  Abs 1013063 #F7547 -   143   117   125
=OptDev  Abs       8 #00008 -    30   412  1491  1495  2071
=PACKp   Abs 1014192 #F79B0 -  1485
=PASSp   Abs 1014643 #F7B73 -  2060
 PASp10  Abs 1014662 #F7B86 -  2069  2064
 PASpER  Abs 1014658 #F7B82 -  2066  2122
 PRNTPE  Abs 1013099 #F756B -   160    72
=PRNTSp  Abs 1012989 #F74FD -    69
 READP5  Ext                -   133
=REMOTp  Abs 1013278 #F761E -   410
=REQSTp  Abs 1014621 #F7B5D -  2014
 REQp10  Abs 1014639 #F7B6F -  2021  2017
=RESETp  Abs 1013288 #F7628 -   453
=RESPTR  Abs 1014555 #F7B1B -  1875   157   232   291   362   510   673   821
                              1373  1392  1427  1500  1507  1510  1818  2018
                              2132
 REST*   Ext                -   562
=RESTp   Abs 1014702 #F7BAE -  2125
=RSDOD1  Abs 1014442 #F7AAA -  1761   354   925  1138  1405  1715
=RTNCC   Abs 1014190 #F79AE -  1434   287   289
 Resptr  Abs 1014720 #F7BC0 -  2132  2130
 SENDP1  Abs 1013452 #F76CC -   777   778   843
 SENDP2  Abs 1013471 #F76DF -   802   808
 SENDP3  Abs 1013499 #F76FB -   821   803
 SENDP4  Abs 1013552 #F7730 -   842   824   832
 SENDP5  Abs 1013559 #F7737 -   849   786   798   818
 SENDlp  Abs 1013485 #F76ED -   814   817
=SENDp   Abs 1013448 #F76C8 -   762
=SKIP    Abs 1014582 #F7B36 -  1915
 ST!NO1  Abs 1013835 #F784B -  1127  1121
 ST!NO2  Abs 1013837 #F784D -  1134  1119  1126
=ST!NOp  Abs 1013804 #F782C -  1110   797   807  2016
=STANDp  Abs 1013197 #F75CD -   284
=STANp+  Abs 1013180 #F75BC -   238   292
=SVDOD1  Abs 1014419 #F7A93 -  1752   350   900  1114  1374  1712
 Skip    Abs 1014579 #F7B33 -  1914  1917
=SpChar  Abs       2 #00002 -    19
=StarOK  Abs      10 #0000A -    26    27   413  1291  1301  1486  1490  2070
=StrOK   Abs      10 #0000A -    27   849  1030  1049  1125  2015
=TRIGp   Abs 1013278 #F761E -   411
 USINGp  Ext                -   126
 Ucrang  Ext                -   975   992
 WRDSCN  Ext                -  2075
```

```
=XWORDp  Abs 1014190 #F79AE -  1433
=XWRD1p  Abs 1013169 #F75B1 -   229
=cATCH+  Abs 1014548 #F7B14 -  1842  1394
 chkOK   Abs 1013087 #F755F -   155   146   149
 eILEXp  Ext                 -  1670  1836
 eILPAr  Ext                 -   669  1300
 eMSPAr  Ext                 -   217  1345
 eSYNTx  Ext                 -   160   244
=oUT1TK  Abs 1014462 #F7ABE -  1773   225   917  1324  1600
=oUT2TC  Abs 1014469 #F7AC5 -  1776  1270
=oUT3TK  Abs 1014479 #F7ACF -  1780   349
=oUTNBS  Abs 1014489 #F7AD9 -  1784   837  1052
 t%      Ext                 -  1318
 t*      Ext                 -   662  1294
 t@      Ext                 -   155
 tCNTRL  Ext                 -  2063
 tCOLON  Ext                 -  1266  2135
 tCOMMA  Ext                 -   358  1115  1502
 tINTRR  Ext                 -   504   607  1963
 tIO     Ext                 -   558
 tIS     Ext                 -    70
 tLITRL  Ext                 -  1235  1407
 tLOCKO  Ext                 -   341
 tOFF    Ext                 -   288  2119
 tON     Ext                 -   286  2117
 tSEMIC  Ext                 -   129   147   902   911  1267  1359
 tUSING  Ext                 -   144
 tXWORD  Ext                 -   339   502   556   605  1961  2061
 wrdscn  Abs 1014675 #F7B93 -  2075   285   501   555   604  1960  2060  2116
```

Input Parameters

  Source file name is NZ&PAR::MS

  Listing file name is NZ/PAR:TI:ML::-1

  Object file name is NZ%PAR:TI:MS::-1

```
                                    111111
                          0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1                *
 2                *       N   N  ZZZZZ   &     DDDD   EEEEE   CCC
 3                *       N   N      Z  & &     D  D  E      C   C
 4                *       NN  N     Z   & &     D  D  E      C
 5                *       N N N    Z     &      D  D  EEEE   C
 6                *       N  NN   Z     & & &   D  D  E      C
 7                *       N   N  Z      &  &    D  D  E      C   C
 8                *       N   N  ZZZZZ   && &  DDDD   EEEEE   CCC
 9                *
10                *
11                        TITLE  PIL DECOMPILE ROUTINES<831027.1220>
12 F7BD3                  ABS    #F7BD3        TI%HP6 address (fixed)
13                ****************************************************************
14                ****************************************************************
15                **
16                ** Name:       PRNTSD - PRINTER IS decompile routine
17                ** Name:       PACKd  - PACK decompile (device spec,OUTELA)
18                **
19                ** Category:   STDCMP
20                **
21                ** Purpose:
22                **      Decompile the PRINTER IS/PACK statements
23                **
24                ** Entry:
25                **      D1 points to tokenized device spec
26                **      D0 points to output buffer
27                **      D[A] is end of available memory, P=0
28                **
29                ** Exit:
30                **      Exits through OUTELA
31                **      Carry clear, P=0
32                **
33                ** Calls:      OUT3TC,?A=CLN,PILDC,?A=CMA,OUTCMA,EXPRDC
34                **
35                ** Uses.......
36                **  Exclusive: A,  C
37                **  Inclusive: A,B,C,R0,R1,R2,D0,D1,P,ST[0,3,8,10,11]
38                **
39                ** Stk lvls:   6 (PILDC)
40                **
41                ** Detail:
42                **      Decompiles 1 or more device specs (separated by
43                **        commas)
44                **
45                ** History:
46                **
47                **    Date      Programmer              Modification
48                **  --------    ----------    ---------------------------------
49                **  12/22/82      NZ          Updated documentation
50                **
51                ****************************************************************
52                ****************************************************************
53 F7BD3          =PRNTSd
54 F7BD3 3594             LCASC  \ SI\          "IS "
         3502
```

```
 55 F7BDB 7000          GOSUB  =OUT3TC       Output 3 tokens!
 56              *
 57              * Device decompile
 58              *
 59 F7BDF 14B   =PACKd  A=DAT1 B             Read in the token (OUT3TC kills)
 60 F7BE2 7003          GOSUB  ?A=CLN        Is this a colon?
 61 F7BE6 571           GONC   PACKD6        No...string expression
 62              *
 63              * D1 points to tCOLON of a device specifier
 64              *
 65 F7BE9 7BB1          GOSUB  PILDC         Decompile the device specifier
 66 F7BED 77E2          GOSUB  ?A=CMA        Is there a comma?
 67 F7BF1 501           GONC   PACKD9        No...exit
 68 F7BF4 171           D1=D1+ 2             Yes...skip it,
 69 F7BF7 7ED1          GOSUB  Outcma          output it, continue
 70 F7BFB 53E           GONC   PACKd         Go always!
 71              *-
 72              *-
 73 F7BFE 7F62 PACKD6   GOSUB  Exprdc        String expression specifier
 74 F7C02 6850 PACKD9   GOTO   Outela        Output End-Of-Line
 75              ********************************************************************
 76              ********************************************************************
 77              **
 78              ** Name:       OUTPd - OUTPUT decompile routine
 79              **
 80              ** Category:   STDCMP
 81              **
 82              ** Purpose:
 83              **      Decompile the OUTPUT statement
 84              **
 85              ** Entry:
 86              **      D0 points to the output buffer
 87              **      D1 points to the input buffer (tokens)
 88              **      D[A] is the end of available memory
 89              **      P=0
 90              **
 91              ** Exit:
 92              **      D0 at next position in output buffer
 93              **      D1 at next character in input buffer
 94              **      P=0
 95              **
 96              ** Calls:      ?A=CLN,PILDC,?A=CMA,OUTCMA,OUTBLK,EXPRDC
 97              **
 98              ** Uses.......
 99              **  Exclusive: A,  C,            D1
100              **  Inclusive: A,B,C,R0,R1,R2,D0,D1,P,ST[0,3,8,10,11]
101              **
102              ** Stk lvls:   6 (PILDC)
103              **
104              ** History:
105              **
106              **    Date     Programmer            Modification
107              **  --------   ----------   ---------------------------------
108              **  12/22/82      NZ        Updated documentation
109              **
```

```
110                *****************************************************************
111                *****************************************************************
112 F7C06          =OUTPd
113 F7C06 7CD2             GOSUB  ?A=CLN
114 F7C0A 572              GONC   OUTPd4        Not COLON: must be string expr
115 F7C0D 7791 OUTPd1      GOSUB  PILDC         Decompile the device spec
116 F7C11          OUTPd2
117 F7C11 73C2             GOSUB  ?A=CMA        A=DAT1 B; LC(2) =tCOMMA
118 F7C15 171              D1=D1+ 2             Skip this token (tCOMMA or t@)
119 F7C18 966              ?A#C   B             Match?
120 F7C1B 90               GOYES  OUTPd3        No...go to DISPDC
121 F7C1D 78B1             GOSUB  Outcma        Yes...output the comma, loop back
122 F7C21 5BE              GONC   OUTPd1        Go always
123                *-
124                *-
125                *
126                * Now have a non-comma token...must be the t@ I added
127                *
128 F7C24 79A1 OUTPd3      GOSUB  Outblk        Output a trailing blank
129 F7C28 14B              A=DAT1 B             Read the next char for DISPDC
130 F7C2B 8D00             GOVLNG =DISPDC       Continue at DISP decompile
            000
131                *-
132                *-
133 F7C32 7B32 OUTPd4      GOSUB  Exprdc        Output the expression
134 F7C36 6ADF             GOTO   OUTPd2        (Token is t@...never comma)
135                *****************************************************************
136                *****************************************************************
137                **
138                ** Name:       INITd - Decompile INITIALIZE statement
139                **
140                ** Category:   STDCMP
141                **
142                ** Purpose:
143                **      Decompile the INITIALIZE statement
144                **
145                ** Entry:
146                **      D0 points to the output buffer
147                **      D1 points to the input buffer
148                **      D[A] is the end of available memory
149                **      P=0
150                **      A[B]=data pointed to by D1
151                **
152                ** Exit:
153                **      D0,D1 positioned after the INITIALIZE statement
154                **      P=0
155                **
156                ** Calls:      OUTNBC,FILDC*,?A=CMA,OUTCMA,EXPRDC
157                **
158                ** Uses.......
159                **  Exclusive: A,  C,            D1,P
160                **  Inclusive: A,B,C,R0,R1,R2,D0,D1,P,ST[0,3,8,10,11]
161                **
162                ** Stk lvls:   6 (FILDC*)
163                **
```

```
164                ** History:
165                **
166                **    Date     Programmer             Modification
167                **  --------   ----------   ---------------------------------
168                **  12/22/82      NZ        Updated documentation
169                **
170                ********************************************************************
171                ********************************************************************
172 F7C3A          =INITd
173 F7C3A 3794     A554           LCASC  \ EZI\        "IZE " OF INITIAL IZE
           02
174                *
175                * Back up the output pointer ("INITIAL " is out already)
176                *
177 F7C44 27               P=     7             Output 8 nibbles (IZE )
178 F7C46 181              D0=D0- 2             Back up over the blank...
179 F7C49 7000             GOSUB  =OUTNBC       Output P+1 nibbles
180 F7C4D 8F00             GOSBVL =FILDC*       Output the file specifier
           000
181 F7C54          INITD0
182 F7C54 7082             GOSUB  ?A=CMA        Is there a tCOMMA?
183 F7C58 4C0              GOC    INITD3        Yes...decompile the expression
184 F7C5B          Outela
185 F7C5B 14B      =XWORDd A=DAT1 B             (Could change to GOVLNG =OUTEL1)
186 F7C5E 8D00             GOVLNG =OUTELA       Output end of line
           000
187                *-
188                *-
189                *
190                * Found an optional parameter expression
191                *
192 F7C65 171      INITD3 D1=D1+ 2             Skip the comma token
193 F7C68 7D61             GOSUB  Outcma        OUTPUT COMMA
194                *
195                * Entry for <XWORD> <Expression> [, <Expression> ]*
196                *
197 F7C6C          =STANd+
198 F7C6C 7102     =INITD2 GOSUB  Exprdc        Decompile the expression
199 F7C70 63EF             GOTO   INITD0        Check if more follows
200                ********************************************************************
201                ********************************************************************
202                **
203                ** Name:       STANDd - STANDBY decompile
204                **
205                ** Category:   STDCMP
206                **
207                ** Purpose:
208                **      Decompile the STANDBY statement
209                **
210                ** Entry:
211                **      D1 points to the tokenized statement
212                **      D0 points to the output buffer
213                **      D[A] is the end of available memory
214                **      P=0
```

```
                **
                ** Exit:
                **      D0, D1 updated past statement contents
                **      P=0
                **
                ** Calls:      LOOP#d,<INITD2>
                **
                ** Uses.......
                **  Inclusive: A,B,C,R0,R1,R2,D0,D1,P,ST[0,3,8,10,11]
                **
                ** Stk lvls:   5 (EXPRDC)
                **
                ** History:
                **
                **    Date     Programmer              Modification
                **  --------   ----------    --------------------------------
                **  02/25/83      NZ         Added documentation
                **
                ****************************************************************
                ****************************************************************
F7C74 77C0 =STANDd GOSUB  LOOP#d        Decompile optional loop #
F7C78 3100         LC(2)  =tON
F7C7C 962          ?A=C   B             Is this STANDBY ON?
F7C7F B0           GOYES  STANdj        Yes...output text
F7C81 3100         LC(2)  =tOFF
F7C85 966          ?A#C   B             Is this STANDBY OFF?
F7C88 4E           GOYES  STANd+        No...must be expression
F7C8A 6712 STANdj  GOTO   CNTRLd        Decompile shared with CONTROL
                ****************************************************************
                ****************************************************************
                **
                ** Name:       LOCALd  -  Decompile LOCAL statement
                **
                ** Category:   STDCMP
                **
                ** Purpose:
                **      Decompile LOCAL [ LOCKOUT ] statement
                **
                ** Entry:
                **      D0 points to the output buffer
                **      D1 points to the input buffer
                **      D[A] is the end of available memory
                **      P=0
                **
                ** Exit:
                **      D0,D1 positioned after the LOCAL statement
                **      P=0
                **
                ** Calls:      GTEXT+,?A=CMA,OUTBLK,EXPRDC
                **
                ** Uses.......
                **  Inclusive: A,B,C,R0,R1,R2,D0,D1,P,ST[0,3,8,9,10,11]
                **
                ** Stk lvls:   5 (EXPRDC)
                **
```

```
                ** History:
                **
                **    Date     Programmer          Modification
                **  --------   ----------   --------------------------------
                **  10/26/83      NZ        Updated documentation
                **  02/01/83      JH        Added Routine
                **
                ********************************************************************
                ********************************************************************
F7C8E 1585      =LOCALd A=DAT1 6
F7C92 AF6               C=A    W            Set high nibs for compare
                *
                * Following lines are REALLY...
                *       LC(6)  (=tLOCKO)~(=LEXPIL)~(tXWORD)
                *****
F7C95 35                NIBHEX 35           LC(6)....
F7C97 00                CON(2) =tXWORD      tXWORD~...
F7C99 00                CON(2) =LEXPIL      LEXPIL~.
F7C9B 00                CON(2) =tLOCKO      tLOCKO.
                *****
F7C9D 976               ?A#C   W            Is this LOCAL LOCKOUT?
F7CA0 72                GOYES  CLEARd       No...just a device specifier
                *
                * LOCAL LOCKOUT...
                *
F7CA2 849               ST=0   9            No trailing blank
F7CA5 8F00              GOSBVL =GTEXT+
      000
F7CAC 7822      Loopd   GOSUB  ?A=CMA       A=DAT1 B;LC(2) =tCOMMA
F7CB0 171               D1=D1+ 2
F7CB3 962               ?A=C   B            Loop specifier?
F7CB6 D0                GOYES  LOCLd1       No...done
F7CB8 1C1               D1=D1- 2            Yes...skip the tCOMMA
F7CBB 7211              GOSUB  Outblk       Output the blank
F7CBF 7EA1              GOSUB  Exprdc       Decompile the loop expression
F7CC3 679F      LOCLd1  GOTO   Outela       Output end of line
                ********************************************************************
                ********************************************************************
                **
                ** Name:       CLEARd, TRIGd, REMOTd - Device spec decompile
                **
                ** Category:   STDCMP
                **
                ** Purpose:
                **      Decompile CLEAR, TRIGGER and REMOTE statements
                **
                ** Entry:
                **      D0 points to the output buffer
                **      D1 points to the input buffer
                **      D[A] is the end of available memory
                **      P=0
                **
                ** Exit:
                **      D0, D1 positioned after the CLEAR, LOCAL or REMOTE stmt
                **      P=0
```

```
                **
                ** Calls:      ?A=CMA,<PACKd>
                **
                ** Uses.......
                **  Inclusive: A,B,C,R0,R1,R2,D0,D1,P,ST[0,3,8,10,11]
                **
                ** Stk lvls:   6 (PILDC)
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **  --------   ----------   --------------------------------
                **  10/26/83      NZ        Updated documentation
                **  02/01/83      JH        Added optional device capability
                **
                ********************************************************************
                ********************************************************************
F7CC7           =REMOTd
F7CC7           =TRIGd
F7CC7           =CLEARd
F7CC7 7D02              GOSUB   ?A=CMA          Check for tCOMMA
F7CCB 590               GONC    CLRD10          Not found...decompile device spec
F7CCE 171               D1=D1+ 2                Found tCOMMA...skip it (EOL)
F7CD1 698F              GOTO    Outela          Output end of line
                *-
                *-
F7CD5           CLRD10
                *
                * Now have a <Device spec>
                *
F7CD5 690F              GOTO    PACKd
                ********************************************************************
                ********************************************************************
                **
                ** Name:       OFFIOd - OFF IO and OFF INTR decompile
                ** Name:       RESTd - RESTORE IO decompile
                **
                ** Category:   STDCMP
                **
                ** Purpose:
                **      Decompile the "IO" part of OFF IO and RESTORE IO
                **      Decompile the "INTR" part of OFF INTR
                **      Decompile the "IO " <expr> of RESTORE IO [<num expr>]
                **
                ** Entry:
                **      D0 points to the output buffer
                **      D1 points to the input buffer (tokenized line)
                **      A[B]=next input token
                **      D[A] is then end of available memory
                **      P=0
                **
                ** Exit:
                **      Exits through OUTELA
                **      D0 points to the output buffer
                **      D1 points to the input buffer
```

```
379            **
380            ** Calls:      OtINTR,IOd,IOdspc
381            **
382            ** Uses.......
383            **  Exclusive:   C
384            **  Inclusive: A,C,D0,D1,P
385            **
386            ** Stk lvls:   3 (IOdspc)
387            **
388            ** History:
389            **
390            **    Date     Programmer             Modification
391            **  --------   ----------   --------------------------------
392            **  12/22/82      NZ        Updated documentation
393            **
394            ****************************************************************
395            ****************************************************************
396 F7CD9 3100 =OFFIOd LC(2)  =tXWORD
397 F7CDD 966          ?A#C   B
398 F7CE0 C0           GOYES  OFIOd1
399            *
400            * This is OFF INTR
401            *
402 F7CE2 175          D1=D1+ 6              Step over the tINTR
403 F7CE5 7DD1         GOSUB  OtINTR         Output the INTR
404 F7CE9 560          GONC   OFIOd2         Go always
405            *-
406            *-
407 F7CEC      OFIOd1
408 F7CEC 7400         GOSUB  IOd            Decompile "IO"
409 F7CF0 6A6F OFIOd2  GOTO   Outela         Exit
410            *-
411            *-
412 F7CF4 3394 IOd     LCASC  \OI\
           F4
413 F7CFA 6000 Out2tc  GOTO   =oUT2TC        Output 2 tokens from C
414            *
415            * Output "IO ", decompile an expression
416            *
417 F7CFE 7BC0 =RESTd  GOSUB  IOdspc         Decompile "IO "
418 F7D02 66A1         GOTO   CNTRL9         Finish up with expression
419            ****************************************************************
420            ****************************************************************
421            **
422            ** Name:       ASGNd - ASSIGN IO decompile
423            **
424            ** Category:   STDCMP
425            **
426            ** Purpose:
427            **      Decompile the ASSIGN IO statement
428            **
429            ** Entry:
430            **      D0 points to the output buffer
431            **      D1 points to the input buffer (tokenized statement)
432            **      D[A] is the end of available memory
```

```
433             **      P=0
434             **
435             ** Exit:
436             **      Exits through PACKd
437             **
438             ** Calls:      IOdspc,<PACKd>
439             **
440             ** Uses.......
441             **  Inclusive: A,B,C,R0,R1,R2,D0,D1,P,ST[0,3,8,10,11]
442             **
443             ** Stk lvls:   5 <PACKd>
444             **
445             ** History:
446             **
447             **    Date     Programmer              Modification
448             **  --------   ----------    --------------------------------
449             **  12/22/82      NZ         Updated documentation
450             **
451             ****************************************************************
452             ****************************************************************
453 F7D06      =ASGNd
454 F7D06 73C0         GOSUB  IOdspc         Decompile "IO "
455 F7D0A 64DE         GOTO   PACKd          Device Decompile!
456             ****************************************************************
457             ****************************************************************
458             **
459             ** Name:       RESETd - RESET HPIL decompile
460             **
461             ** Category:   STDCMP
462             **
463             ** Purpose:
464             **      Decompile the RESET HPIL statement
465             **
466             ** Entry:
467             **      D1 points past the RESET token
468             **      D0 points to the output buffer
469             **      D[A] is the end of available memory
470             **      P=0
471             **
472             ** Exit:
473             **      Output buffer has "RESET HPIL"
474             **      D0, D1 past the statement
475             **
476             ** Calls:      OUTNBC,<Loopd>
477             **
478             ** Uses.......
479             **  Inclusive: A,B,C,D0,D1,R0,R1,R2,P,ST[0,3,8,10,11]
480             **
481             ** Stk lvls:   5 (Loopd)
482             **
483             ** History:
484             **
485             **    Date     Programmer              Modification
486             **  --------   ----------    --------------------------------
487             **  02/18/83      NZ         Added loop number decompile
```

```
488                ** 12/22/82      NZ          Updated documentation
489                **
490                ****************************************************************
491                ****************************************************************
492 F7D0E 3784     =RESETd LCASC  \LIPH\
          0594
          C4
493 F7D18 27             P=     7
494 F7D1A 7000           GOSUB  =OUTNBC       Output "HPIL"
495 F7D1E 6D8F           GOTO   Loopd
496                ****************************************************************
497                ****************************************************************
498                **
499                ** Name:       SENDd - Decompile the SEND statement
500                **
501                ** Category:   STDCMP
502                **
503                ** Purpose:
504                **      Decompile the SEND statement (also works for ENABLE
505                **      INTR and REQUEST)
506                **
507                ** Entry:
508                **      D1 points to the first item following the SEND token
509                **      D0 points to the output buffer
510                **      D[A] is the end of available memory
511                **      A[B] is the next token (at D1)
512                **      P=0
513                **
514                ** Exit:
515                **      D0,D1 after SEND command, P=0
516                **      Exits through OUTELA
517                **
518                ** Calls:      LOOP#d,FRASPd,ST!NOd,<OUTELA>
519                **
520                ** Uses.......
521                **  Inclusive: A,B,C,R0,R1,R2,D0,D1,P,ST[0,3,8,10,11]
522                **
523                ** Stk lvls:   6 (LOOP#d)(ST!NOd)
524                **
525                ** History:
526                **
527                **    Date      Programmer              Modification
528                **  --------    ----------    --------------------------------
529                **  12/22/82      NZ          Updated documentation
530                **
531                ****************************************************************
532                ****************************************************************
533                *
534                * SEND decompile will also work for REQUEST and ENABLE INTR
535                *
536 F7D22 70A1     =ENABLd GOSUB  OtINTR        Decompile "INTR "
537 F7D26 14B            A=DAT1 B              Read in the next token
538 F7D29        =REQSTd
539 F7D29 7210     =SENDd  GOSUB  LOOP#d
540                *
```

```
                * LOOP#d decompiles the loop number, if any, and returns with
                * A[B] containing the next token
                *
                * FRASPD decompiles a frame spec, if any. If not a frame spec,
                * it returns with carry set. In either case, A[B] is the next
                * token.
                *
F7D2D 7D20      SENDD1  GOSUB  FRASPd
F7D31 5BF               GONC   SENDD1        Loop until frame spec not found
                *
                * If here, either EOL or expression
                *
                * ST!NOd Decompiles the string or numeric expression(s), if
                * any. If none are found, it returns with carry set.
                *
F7D34 7B40              GOSUB  ST!NOd
F7D38 54F               GONC   SENDD1        Continue with next frame spec
                *
                * If here, have reached end-of-line
                *
F7D3B 6F1F              GOTO   Outela        Output end of line
                ****************************************************************
                ****************************************************************
                **
                ** Name:       LOOP#d - Decompile an optional loop #
                **
                ** Category:   DCMUTL
                **
                ** Purpose:
                **      Decompile a loop number, if any. If none present, exit
                **      with carry set (Leaves next token in A[B])
                **
                ** Entry:
                **      D1 points to the (optional) loop #
                **      D0 points to the output buffer
                **      D[A] is the end of available memory
                **      A[B] is the next token (at D1)
                **
                ** Exit:
                **      D0,D1 positioned after the loop #, if found
                **      A[B] is the next token
                **      Carry set if no loop #, clear if loop # found
                **
                ** Calls:      EXPDC+,OUT2TC
                **
                ** Uses.......
                **  Exclusive: A,  C,            D1
                **  Inclusive: A,B,C,R0,R1,R2,D0,D1,P,ST[0,3,8,10,11]
                **
                ** Stk lvls:   5 (EXPDC+)
                **
                ** History:
                **
                **    Date     Programmer              Modification
                **  --------   ----------    ---------------------------------
```

```
596                ** 03/01/83      NZ          Updated to read token after expr
597                ** 12/22/82      NZ          Updated documentation
598                **
599                ********************************************************************
600                ********************************************************************
601 F7D3F          =LOOP#d
602 F7D3F 3100             LC(2)  =tSEMIC
603 F7D43 966              ?A#C   B
604 F7D46 00               RTNYES                 Not a loop #...return, carry set
605 F7D48 7221             GOSUB  Expdc+          Expression decompile
606 F7D4C 33B3             LCASC  \ ;\
          02
607 F7D52 74AF             GOSUB  Out2tc          Output terminating <semic><blank>
608 F7D56 171              D1=D1+ 2               Skip tSEMIC following the expr
609 F7D59 14B              A=DAT1 B               Read next token
610 F7D5C 03               RTNCC                  Return, carry clear (LOOP #)
611                ********************************************************************
612                ********************************************************************
613                **
614                ** Name:        FRASPd - Decompile a frame spec
615                **
616                ** Category:    DCMUTL
617                **
618                ** Purpose:
619                **      Frame spec decompile routine
620                **
621                ** Entry:
622                **      D0 points to the output buffer
623                **      D1 points to the input buffer (tokens)
624                **      D[A] is the end of available memory
625                **      A[B] is the next token (at D1)
626                **      P=0
627                **
628                ** Exit:
629                **      A[B] is next token
630                **      Carry clear if frame spec found, set if not found
631                **      D0,D1 updated to current position
632                **
633                ** Calls:       ?A=CLN,OUT1TK,RANGER,Outblk
634                **
635                ** Uses.......
636                ** Exclusive:  A,C,   D1
637                ** Inclusive:  A,C,D0,D1
638                **
639                ** Stk lvls:   2 (OUT1TK)(Outblk)
640                **
641                ** History:
642                **
643                **    Date      Programmer              Modification
644                **  --------    ----------    ---------------------------------
645                **  12/22/82      NZ          Updated documentation
646                **
647                ********************************************************************
648                ********************************************************************
649 F7D5E          =FRASPd
```

```
650 F7D5E 7481          GOSUB  ?A=CLN
651 F7D62 480           GOC    FRASd2          This is a frame spec (Skip COLON)
652 F7D65 02            RTNSC                  Not a frame (return, carry set)
653             *-
654             *-
655 F7D67 7000 FRASd1   GOSUB  =oUT1TK         Output the character
656 F7D6B 171  FRASd2   D1=D1+ 2               Skip the current token/character
657 F7D6E 14B           A=DAT1 B               Read next character
658 F7D71 8E00          GOSUBL =RANGEA         Check if in [A-Z]
          00
659 F7D77 5FE           GONC   FRASd1          Yes...continue
660             *
661             * Output a trailing blank after mnemonic
662             *
663 F7D7A 7350          GOSUB  Outblk
664 F7D7E AEE           ACEX   B               Restore item (OUTBYT does ACEX)
665 F7D81 03            RTNCC                  End of frame (return, carry clear)
666             ****************************************************************
667             ****************************************************************
668             **
669             ** Name:       ST!NOd - Decompile a string or numeric expr
670             **
671             ** Category:   DCMUTL
672             **
673             ** Purpose:
674             **      Decompile string or numeric expr (Preceded by tCOMMA)
675             **
676             ** Entry:
677             **      D0 points to the output buffer
678             **      D1 points to the input buffer (tokens)
679             **      D[A] is the end of available memory
680             **      A[B] is the next token (at D1)
681             **      P=0
682             **
683             ** Exit:
684             **      A[B] is next token, D1 points to next token
685             **      D0, D1 updated to current position, P=0
686             **      Carry set if not a string or a numeric expression
687             **
688             ** Calls:      EXPDC+,?A=CM+,Outcma,Outblk
689             **
690             ** Uses.......
691             **  Exclusive: A,  C,            D1
692             **  Inclusive: A,B,C,R0,R1,R2,D0,D1,P,ST[0,3,8,10,11]
693             **
694             ** Stk lvls:   5 (EXPDC+)
695             **
696             ** History:
697             **
698             **    Date     Programmer         .     Modification
699             **  --------   ----------     ---------------------------------
700             **  12/22/82      NZ          Updated documentation
701             **
702             ****************************************************************
703             ****************************************************************
```

```
F7D83 3100 =ST!NOd LC(2)  =tCOMMA
F7D87 966          ?A#C   B
F7D8A 00           RTNYES                 Not an expression (RTNSC)
F7D8C 7ED0 ST!Nd1  GOSUB  Expdc+          D1=D1+2;EXPRDC
           *
           * A[B] is next item
           *
F7D90 7741         GOSUB  ?A=CM+
F7D94 5A0          GONC   ST!Nd2          Done with expression list...exit
           *
           * Another expression follows
           *
F7D97 7E30         GOSUB  Outcma          Output a comma between items
F7D9B 60FF         GOTO   ST!Nd1          Loop back and continue
           *-
           *-
F7D9F 7E20 ST!Nd2  GOSUB  Outblk          (Saves A[B] in C[B])
F7DA3 AEE          ACEX   B               Restore item from C[B]
F7DA6 03           RTNCC                  Exit, carry clear
           ****************************************************************
           ****************************************************************
           **
           ** Name:       PILDC - Decompile an HPIL device specifier
           **
           ** Category:   DCMUTL
           **
           ** Purpose:
           **      Decompile an HP-IL device spec stored as a literal:
           **      case:
           **       <t*>
           **       or <t%><numeric expression>[( <numeric expression> )]
           **       or <numeric expression>
           **       or <tLITRL> <literal> [( <numeric expression> )]
           **       or <tSEMIC> <volume label>
           **
           ** Entry:
           **      D1 points to the tCOLON in the input buffer
           **      D0 points to the output buffer
           **      D[A] is the end of available memory
           **      P=0
           **
           ** Exit:
           **      D0 points after the last character of the output line
           **      D1 points to the first token following the input tokens
           **      P=0
           **
           ** Calls:      OUTBYT,EXPDC+,?A=CLN,OUT1TK,EXPRDC
           **
           ** Uses.......
           **  Exclusive: A,  C,          D0,D1
           **  Inclusive: A,B,C,R0,R1,R2,D0,D1,P,ST[0,3,8,10,11]
           **
           ** Stk lvls:   5 (EXPRDC)(EXPDC+)
           **
           ** History:
```

```
                **
                **    Date     Programmer          Modification
                ** --------   ----------   --------------------------------
                ** 12/22/82      NZ        Updated documentation
                **
                ********************************************************************
                ********************************************************************
                *
                * Syntax:
                *   Input stream:
                *       <t*>
                *   or  <t%> <num expr> [ <tCOLON> <num expr> ]
                *   or  <num expr>
                *   or  <tLITRL> <literal data> [ <tCOLON> <num expr> ]
                *   or  <tSEMIC> <literal volume label>
                *
                *   Output text:
                *       *
                *   or  :%<num expr> [ (<num expr>) ]
                *   or  :<num expr>
                *   or  :<literal data> [ (<num expr>) ]
                *   or  .<volume label>
                *
F7DA8 31A3 =PILDC  LCASC  \:\
F7DAC 7910         GOSUB  Outbyt        Output the colon
F7DB0 171          D1=D1+ 2
F7DB3 14B          A=DAT1 B             Read the next token
                *
                * Check for "*" token
                *
F7DB6 3100         LC(2)  =t*
F7DBA 966          ?A#C   B             Is it t*?
F7DBD 42           GOYES  PILDC2        No...check further
F7DBF 181          D0=D0- 2             Yes...undo the ":"
F7DC2 171          D1=D1+ 2             Skip the "*" token
F7DC5 31A2         LCASC  \*\
F7DC9 6000 Outbyt  GOTO   =OUTBYT       Done with this device spec
                *-
                *-
F7DCD 732F IOdspc  GOSUB  IOd           Output "IO "
F7DD1 3102 Outblk  LCASC  \ \
F7DD5 63FF         GOTO   Outbyt
                *-
                *-
F7DD9 31C2 Outcma  LCASC  \,\
F7DDD 6BEF         GOTO   Outbyt
                *-
                *-
F7DE1 3100 PILDC2  LC(2)  =t%
F7DE5 966          ?A#C   B             Is it Accessory ID?
F7DE8 F2           GOYES  PILDC5        No...check further
F7DEA 3152         LCASC  \%\           Yes...output %
                *
                * Accessory ID
                *
```

```
814 F7DEE 77DF PILDC3  GOSUB  Outbyt
815 F7DF2 7870         GOSUB  Expdc+        Step over t% first
816 F7DF6 7CE0         GOSUB  ?A=CLN        "(" token kludge
817 F7DFA 506          GONC   PILDC9        Not "(" token...check loop #
818 F7DFD 3182 PILDC4  LCASC  \(\
819 F7E01 74CF         GOSUB  Outbyt
820 F7E05 7560         GOSUB  Expdc+        (Step over tCOLON first)
821 F7E09 3192         LCASC  \)\
822 F7E0D 78BF         GOSUB  Outbyt        Send the closing ")"
823 F7E11 AEE          ACEX   B             Get token back to A[B]
824 F7E14 564          GONC   PILDC9        Go always to check for loop #
825             *-
826             *-
827             *
828             * Not Accessory ID - perhaps a device word
829             *
830 F7E17 3100 PILDC5  LC(2)  =tLITRL
831 F7E1B 966          ?A#C   B             Is this a literal?
832 F7E1E 42           GOYES  PILDC8        No...must be an address expression
833 F7E20 171          D1=D1+ 2             Skip =tLITRL
834             *
835             * If here, this is a literal (device word or Device ID)
836             *
837 F7E23 14B  PILDC6   A=DAT1 B             Read next character
838 F7E26 D6           C=A    A             Copy A[B] to C[B]
839 F7E28 A66          C=C+C  B             If carry, end of literal
840 F7E2B 4C0          GOC    PILDC7        Carry...end of literal
841 F7E2E 171  PILDc6  D1=D1+ 2             Still part of literal...skip input
842             *
843             * Output the character and loop back for next character
844             *
845 F7E31 7000         GOSUB  =oUT1TK       Output from A[B]
846 F7E35 5DE          GONC   PILDC6        Go always - loop back again
847             *-
848             *-
849             *
850             * High bit set...end of literal characters
851             *
852 F7E38       PILDC7
853 F7E38 7AA0         GOSUB  ?A=CLN        Is there a tCOLON ("(")?
854 F7E3C 40C          GOC    PILDC4        Yes...process the expression
855 F7E3F 5B1          GONC   PILDC9        Go always to check loop #
856             *-
857             *-
858 F7E42 3100 PILDC8  LC(2)  =tSEMIC
859 F7E46 966          ?A#C   B             Is this a volume label?
860 F7E49 E0           GOYES  PILDc8        No...must be address expression
861             *
862             * Literal volume label
863             *
864 F7E4B 181          D0=D0- 2             Back over the \:\
865 F7E4E 31E2         LCASC  \.\
866 F7E52 DA           A=C    A             Write out the \.\, then vol label
867 F7E54 59D          GONC   PILDc6        Go always
868             *-
```

```
                        *-
                        *
                        * If here, this must be an address expression
                        *
F7E57 7610 PILDc8  GOSUB  Exprdc
F7E5B 3100 PILDC9  LC(2)  =tSEMIC       Check if there is a loop spec
F7E5F 962          ?A=C   B             Loop specifier?
F7E62 40           GOYES  PILDC!        Yes...process it
F7E64 03           RTNCC                No...return with carry clear
                        *-
                        *-
F7E66 31A3 PILDC!  LCASC  \:\           Loop specifier....
F7E6A 7B5F         GOSUB  Outbyt          output the colon,
F7E6E 171  Expdc+  D1=D1+ 2               then the expression
F7E71 8D00 Exprdc  GOVLNG =EXPRDC
      000
                        ********************************************************************
                        ********************************************************************
                        **
                        ** Name:       PASSd - PASS CONTROL decompile
                        **
                        ** Category:   STDCMP
                        **
                        ** Purpose:
                        **      Decompile the PASS CONTROL statement
                        **
                        ** Entry:
                        **      D1 points to the input buffer (tokens)
                        **      D0 points to the output buffer
                        **      D[A] is the end of available memory
                        **      A[B] is the next token (at D1)
                        **      P=0
                        **
                        ** Exit:
                        **      D0, D1 are positioned after the output/input tokens
                        **      Exits through OUTELA
                        **
                        ** Calls:      OUTNBC,?A=CMA,<PACKd>
                        **
                        ** Uses.......
                        **  Inclusive: A,B,C,R0,R1,R2,D0,D1,P,ST[0,3,8,10,11]
                        **
                        ** Stk lvls:   6 (PACKd)
                        **
                        ** History:
                        **
                        **    Date     Programmer              Modification
                        **   --------   ----------    ---------------------------------
                        **   10/27/83      NZ         Added documentation
                        **
                        ********************************************************************
                        ********************************************************************
F7E78 3F34 =PASSd  LCASC  \ LORTNOC\
      F4E4
      4525
```

```
            F4C4
            02
    921 F7E8A 2F            P=      15
    922 F7E8C 7000          GOSUB   =OUTNBC
    923 F7E90 7440          GOSUB   ?A=CMA
    924 F7E94 590           GONC    PASd10
    925 F7E97 171           D1=D1+  2
    926 F7E9A 60CD OUtela   GOTO    Outela
    927               *-
    928               *-
    929 F7E9E 604D PASd10   GOTO    PACKd
    930               ********************************************************************
    931               ********************************************************************
    932               **
    933               ** Name:        CNTRLd - CONTROL ON/OFF decompile
    934               **
    935               ** Category:    STDCMP
    936               **
    937               ** Purpose:
    938               **      Decompile the CONTROL ON/OFF statements
    939               **
    940               ** Entry:
    941               **      D0 is points to the input buffer (tokens)
    942               **      D1 points to the output buffer
    943               **      D[A] is the end of available memory
    944               **      A[B] is the next token (at D1)
    945               **      P=0
    946               **
    947               ** Exit:
    948               **      D0, D1 positioned after the statement
    949               **      Exits through PACKD6/OUTELA
    950               **
    951               ** Calls:       GTXT++,<OUTELA>,<PACKD6>
    952               **
    953               ** Uses.......
    954               **  Inclusive: A,B,C,R0,R1,R2,D0,D1,P,ST[0,3,8,10,11]
    955               **
    956               ** Stk lvls:   5 (PACKD6)
    957               **
    958               ** History:
    959               **
    960               **    Date      Programmer              Modification
    961               **  --------    ----------    --------------------------------
    962               **  10/27/83       NZ         Added documentation
    963               **
    964               ********************************************************************
    965               ********************************************************************
    966 F7EA2 8F00 =CNTRLd GOSBVL =GTXT++          Output ON/OFF (blanks)
            000
    967 F7EA9 14F  CNTRL9  C=DAT1 B                Check if at end of line
    968 F7EAC 80D1         P=C    1
    969 F7EB0 0C           P=P+1                   If carry, at end of line now
    970 F7EB2 20           P=     0                Reset P=0 regardless
    971 F7EB4 45E          GOC    OUtela           End of line if carry
    972 F7EB7 664D         GOTO   PACKD6
```

```
 973                    ****************************************************************
 974                    ****************************************************************
 975                    **
 976                    ** Name:       ONINTd - ON INTR decompile
 977                    **
 978                    ** Category:   STDCMP
 979                    **
 980                    ** Purpose:
 981                    **      Decompile the ON INTR statement
 982                    **
 983                    ** Entry:
 984                    **      D0 points to the input buffer (tokens)
 985                    **      D1 points to the output buffer
 986                    **      D[A] is the end of available memory
 987                    **      A[B] is the next token (at D1)
 988                    **      P=0
 989                    **
 990                    ** Exit:
 991                    **      D0, D1 positioned after the statement
 992                    **      Exits through ONDC20 (mainframe)
 993                    **
 994                    ** Calls:      OtINTR,<ONDC20>
 995                    **
 996                    ** Uses.......
 997                    **  Inclusive: Same as ONDC20
 998                    **
 999                    ** Stk lvls:   Same as ONDC20
1000                    **
1001                    ** History:
1002                    **
1003                    **    Date      Programmer              Modification
1004                    **  --------    ----------    ---------------------------------
1005                    **  10/27/83       NZ         Added documentation
1006                    **
1007                    ****************************************************************
1008                    ****************************************************************
1009 F7EBB 7700 =ONINTd GOSUB  OtINTR
1010 F7EBF 8D00         GOVLNG =ONDC20       Continue with ON ... GOTO/GOSUB
           000
1011                    *-
1012                    *-
1013                    *
1014                    * Output \INTR\
1015                    *
1016 F7EC6 3994 OtINTR  LCASC  \ RTNI\
           E445
           2502
1017 F7ED2 29           P=     9
1018 F7ED4 6000         GOTO   =OUTNBC       (Returns with P=0)
1019                    *-
1020                    *-
1021                    *
1022                    * Check if A[B] is a tCOMMA (Carry set if so)
1023                    *
1024 F7ED8 14B  =?A=CMA A=DAT1 B
```

```
1025 F7EDB 3100 =?A=CM+ LC(2)  =tCOMMA
1026 F7EDF 962          ?A=C   B
1027 F7EE2 00           RTNYES
1028 F7EE4 01           RTN
1029            *-
1030            *-
1031            *
1032            * Check if A[B] is tCOLON (Carry set if so)
1033            *
1034 F7EE6 3100 =?A=CLN LC(2)  =tCOLON
1035 F7EEA 962          ?A=C   B
1036 F7EED 00           RTNYES
1037 F7EEF 01           RTN
1038 F7EF1              END
```

```
=?A=CLN  Abs 1015526 #F7EE6 -  1034   60  113  650  816  853
=?A=CM+  Abs 1015515 #F7ED8 -  1025  711
=?A=CMA  Abs 1015512 #F7ED8 -  1024   66  117  182  297  344  923
=ASGNd   Abs 1015046 #F7D06 -   453
=CLEARd  Abs 1014983 #F7CC7 -   343  291
 CLRD10  Abs 1014997 #F7CD5 -   350  345
 CNTRL9  Abs 1015465 #F7EA9 -   967  418
=CNTRLd  Abs 1015458 #F7EA2 -   966  242
 DISPDC  Ext                -   130
=ENABLd  Abs 1015074 #F7D22 -   536
 EXPRDC  Ext                -   883
 Expdc+  Abs 1015406 #F7E6E -   882  605  707  815  820
 Exprdc  Abs 1015409 #F7E71 -   883   73  133  198  303  873
 FILDC*  Ext                -   180
=FRASPd  Abs 1015134 #F7D5E -   649  548
 FRASd1  Abs 1015143 #F7D67 -   655  659
 FRASd2  Abs 1015147 #F7D6B -   656  651
 GTEXT+  Ext                -   296
 GTXT++  Ext                -   966
 INITD0  Abs 1014868 #F7C54 -   181  199
=INITD2  Abs 1014892 #F7C6C -   198
 INITD3  Abs 1014885 #F7C65 -   192  183
=INITd   Abs 1014842 #F7C3A -   172
 IOd     Abs 1015028 #F7CF4 -   412  408  798
 IOdspc  Abs 1015245 #F7DCD -   798  417  454
 LEXPIL  Ext                -   287
=LOCALd  Abs 1014926 #F7C8E -   279
 LOCLd1  Abs 1014979 #F7CC3 -   304  300
=LOOP#d  Abs 1015103 #F7D3F -   601  235  539
 Loopd   Abs 1014956 #F7CAC -   297  495
=OFFIOd  Abs 1015001 #F7CD9 -   396
 OFIOd1  Abs 1015020 #F7CEC -   407  398
 OFIOd2  Abs 1015024 #F7CF0 -   409  404
 ONDC20  Ext                -  1010
=ONINTd  Abs 1015483 #F7EBB -  1009
 OUT3TC  Ext                -    55
 OUTBYT  Ext                -   795
 OUTELA  Ext                -   186
 OUTNBC  Ext                -   179  494  922 1018
=OUTPd   Abs 1014790 #F7C06 -   112
 OUTPd1  Abs 1014797 #F7C0D -   115  122
 OUTPd2  Abs 1014801 #F7C11 -   116  134
 OUTPd3  Abs 1014820 #F7C24 -   128  120
 OUTPd4  Abs 1014834 #F7C32 -   133  114
 OUtela  Abs 1015450 #F7E9A -   926  971
 OtINTR  Abs 1015494 #F7EC6 -  1016  403  536 1009
 Out2tc  Abs 1015034 #F7CFA -   413  607
 Outblk  Abs 1015249 #F7DD1 -   799  128  302  663  720
 Outbyt  Abs 1015241 #F7DC9 -   795  783  800  804  814  819  822  881
 Outcma  Abs 1015257 #F7DD9 -   803   69  121  193  716
 Outela  Abs 1014875 #F7C5B -   184   74  304  347  409  561  926
 PACKD6  Abs 1014782 #F7BFE -    73   61  972
 PACKD9  Abs 1014786 #F7C02 -    74   67
=PACKd   Abs 1014751 #F7BDF -    59   70  354  455  929
=PASSd   Abs 1015416 #F7E78 -   920
```

```
 PASd10  Abs 1015454 #F7E9E -   929   924
=PILDC   Abs 1015208 #F7DA8 -   782    65   115
 PILDC!  Abs 1015398 #F7E66 -   880   876
 PILDC2  Abs 1015265 #F7DE1 -   807   791
 PILDC3  Abs 1015278 #F7DEE -   814
 PILDC4  Abs 1015293 #F7DFD -   818   854
 PILDC5  Abs 1015319 #F7E17 -   830   809
 PILDC6  Abs 1015331 #F7E23 -   837   846
 PILDC7  Abs 1015352 #F7E38 -   852   840
 PILDC8  Abs 1015362 #F7E42 -   858   832
 PILDC9  Abs 1015387 #F7E5B -   874   817   824   855
 PILDc6  Abs 1015342 #F7E2E -   841   867
 PILDc8  Abs 1015383 #F7E57 -   873   860
=PRNTSd  Abs 1014739 #F7BD3 -    53
 RANGEA  Ext                -   658
=REMOTd  Abs 1014983 #F7CC7 -   341
=REQSTd  Abs 1015081 #F7D29 -   538
=RESETd  Abs 1015054 #F7D0E -   492
=RESTd   Abs 1015038 #F7CFE -   417
 SENDD1  Abs 1015085 #F7D2D -   548   549   557
=SENDd   Abs 1015081 #F7D29 -   539
=ST!NOd  Abs 1015171 #F7D83 -   704   556
 ST!Nd1  Abs 1015180 #F7D8C -   707   717
 ST!Nd2  Abs 1015199 #F7D9F -   720   712
=STANDd  Abs 1014900 #F7C74 -   235
=STANd+  Abs 1014892 #F7C6C -   197   241
 STANdj  Abs 1014922 #F7C8A -   242   238
=TRIGd   Abs 1014983 #F7CC7 -   342
=XWORDd  Abs 1014875 #F7C5B -   185
 oUT1TK  Ext                -   655   845
 oUT2TC  Ext                -   413
 t%      Ext                -   807
 t*      Ext                -   789
 tCOLON  Ext                -  1034
 tCOMMA  Ext                -   704  1025
 tLITRL  Ext                -   830
 tLOCKO  Ext                -   288
 tOFF    Ext                -   239
 tON     Ext                -   236
 tSEMIC  Ext                -   602   858   874
 tXWORD  Ext                -   286   396
```

Input Parameters

Source file name is NZ&DEC::MS

Listing file name is NZ/DEC:TI:ML::-1

Object file name is NZ%DEC:TI:MS::-1

```
                                111111
                0123456789012345
```

Initial flag settings are

Errors

None

Saturn Assembler News

```
1           *
2           *
3           *       N   N  ZZZZZ   &      SSS   Y   Y  M   M
4           *       N   N      Z  & &    S   S  Y   Y  MM MM
5           *       NN  N     Z   & &    S       Y Y   M M M
6           *       N N N    Z     &      SSS     Y    M M M
7           *       N  NN   Z     & & &      S    Y    M   M
8           *       N   N  Z      &  &   S   S    Y    M   M
9           *       N   N  ZZZZZ   && &   SSS     Y    M   M
10          *
11                  TITLE  Symbolic Assignments <831220.1633>
12          *
13          * Status bit for ATTN key pressed (or other exception cause)
14          *
15          =Attn   EQU    12
16          *
17          * Other status bits
18          *
19          =sPRIVT EQU    11           Status for PRIVATE/SECURE stmt
20          =sUNSEC EQU    10           Status for [UN]Secure statement
21          =sOVERW EQU    8            Status for overwrite existing file
22          =sDevOK EQU    8            Status for device spec exec OK
23          =sSTK   EQU    7            Status for reading from stack
24          =CkTape EQU    5            Status to check for tape device
25          =sLoop? EQU    5            Status for allowing LOOP spec
26          =sReadd EQU    4            Status to force readdress the loop
27          =sFirst EQU    0            Status for first char in filespec
28          *
29          * Status bit corresponding to the bit DIAMOND sets if SREQ?
30          *
31          =sDIAsr EQU    1
32          *
33          * See NZ&PAR for parse status bits
34          *
35          *---------------------------------------------------------------
36          *
37          * Equates for P=, DDL/DDT
38          *
39          * DDL's
40          *
41          =Write0 EQU    0            Write to buffer 0
42          =Write1 EQU    1            Write to buffer 1
43          =Write  EQU    2            Write to tape
44          =SetBP  EQU    3            Set byte pointer
45          =Seek   EQU    4            Seek a record
46          =Format EQU    5            Format the medium
47          =PWrite EQU    6            Partial write mode
48          =Rewind EQU    7            Rewind
49          =CloseR EQU    8            Close record
50          =Xfr01L EQU    9            Transfer buffer 0-->1 (Listener)
51          =XchgL  EQU    10           Exchange buffers 0,1 (Listener)
52          =Verify EQU    11           Verify the medium
53          *
54          * DDT's
55          *
```

```
 56           =Read0  EQU   0           Read from buffer 0
 57           =Read1  EQU   1           Read from buffer 1
 58           =Read   EQU   2           Read from tape
 59           =Positn EQU   3           Read current position
 60           =XchgT  EQU   4           Exchange buffers 0,1 (Talker)
 61           =Xfr01T EQU   5           Transfer buffer 0-->1 (Talker)
 62           =ImpByt EQU   6           Send implementation bytes
 63           =MaxRec EQU   7           Send max addressable record
 64           *
 65           *-------------------------------------------------------------
 66           *
 67           * Equates for device specifiers
 68           *
 69           =DevTyp EQU   #1F         Device type
 70           =DevID  EQU   #3F         Device ID
 71           =VolLbl EQU   #5F         Volume label
 72           =Null   EQU   #7F         "NULL" device
 73           =Loop   EQU   #9F         "LOOP" device
 74           *
 75           *-------------------------------------------------------------
 76           *
 77           * Equates for D[S] values returning from START
 78           *
 79           =DsAddr EQU   0           Device address
 80           =DsDevT EQU   1           Device type
 81           =DsDevI EQU   2           Device ID
 82           =DsVolL EQU   3           Volume label
 83           =DsNull EQU   4           NULL
 84           =DsLoop EQU   5           LOOP
 85           *
 86           *-------------------------------------------------------------
 87           *
 88           * Equates for STANDBY defaults
 89           *
 90           =#Timeo EQU   30          Default # IDY timeouts
 91           =Timout EQU   2*1000      Default timeout between IDY (ms)
 92           *
 93           *-------------------------------------------------------------
 94           *
 95           * PRINT class equate (for OUTPUT class)
 96           *
 97           =OUTPTt EQU   2           This is next after PRINTt
 98           =PLOTt  EQU   (OUTPTt)+1  This is for the PLOT class
 99           *
100           *-------------------------------------------------------------
101           *
102           * I/O buffer numbers - See TI&EQU
103           *
104           *-------------------------------------------------------------
105           *
106           * HPIL frame types (return from FRAME)
107           *
108           =pACK   EQU   00          Acknowledge frame
109           =pSTATE EQU   01          Current Diamond state
110           =pDIAGR EQU   02          Diagnostic test results
```

```
111             =pDIAGL EQU   03          Diagnostic data
112             =pADDR  EQU   04          Address frame
113             =pIFC   EQU   05          IFC received (not active controller)
114             =pEOT   EQU   06          EOT received as controller
115             =pHALTD EQU   07          Conversation halted by Diamond
116             =pTERM  EQU   08          Terminator match (Diamond)
117             =pETE   EQU   09          ETE received
118             =pUTYPE EQU   10          Unrecognized mailbox message type
119             =pDATA  EQU   11          DATA/END frame
120             =pCMD   EQU   12          Command reveived
121             =pRDY   EQU   13          Ready frame reveived
122             =pIDY   EQU   14          IDY reveived
123             =p3DATA EQU   15          Triple byte data
124             *
125             *-------------------------------------------------------
126             *
127             * ERROR TYPES: (See NZ&ERR for most error numbers)
128             *
129             =ePARSE EQU   00          Parse error
130             =eTAPE  EQU   01          Tape error (mass storage error)
131             =ePIL   EQU   02          HPIL error (loop or Diamond)
132             *
133             *-------------------------------------------------------
134             *
135             * Parameters for File Information Buffers (FIB)
136             * See TI&EQU for values and names
137             *
138             *-------------------------------------------------------
139             *
140             * Status bits (for Diamond state)
141             *
142             =sLOCKD EQU   11          Locked out mode (remote)
143             =sRMOTE EQU   10          Remote mode
144             =sDATAO EQU   9           Data in output buffer
145             =sDATAV EQU   8           Data available
146             =sSTAND EQU   7           Controller standby mode
147             =sPOLLE EQU   6           Serial Poll Enabled
148             =sUNCNF EQU   5           Loop is not configured
149             =sINTR  EQU   4           Interrupt pending
150             =sSCNTR EQU   3           System Controller
151             =sTALKA EQU   2           Talker active
152             =sLISTR EQU   1           Listener
153             =sCONTR EQU   0           Controller
154             *
155             *-------------------------------------------------------
156             *
157             * Handshake bits (Diamond to Saturn) (in ST[3:0])
158             *
159             =s3BYTE EQU   3           Triple data byte transfer
160             =sMANUL EQU   2           Diamond is in manual mode
161             =sSRQIN EQU   1           SRQ received on loop
162             =sERROR EQU   0           Error detected/occurred
163             *
164             *-------------------------------------------------------
165             *
```

```
166             * Handshake bits (Diamond to Saturn) (in ST[7:0])
167             *
168             =hs3BYT EQU    7            Triple data xfer
169             =hsMANL EQU    6            Manual mode
170             =hsLPRQ EQU    5            SRQ received from loop
171             =hsERRO EQU    4            Error occured
172             =hsRQSR EQU    3            Diamond SRQ on Saturn Bus
173             =hsAWKE EQU    2            Saturn awake
174             =hsNRD  EQU    1            Saturn NRD
175             =hsMGAV EQU    0            Diamond message available
176             *
177             *---------------------------------------------------------------
178             *
179             * Mailbox opcodes (TO Diamond)
180             *
181             * Frame class
182             *
183             =mFRAME EQU    #1000        Any of the class "FRAME"
184             =mDATAf EQU    #1000        DATA frame
185             =mDATA2 EQU    (mDATAf)/#100
186             =mENDf  EQU    #1200        END frame
187             =mCMDf  EQU    #1400        CoMmanD frame
188             =mCMD3  EQU    (mCMDf)/#10
189             =mCMD2  EQU    (mCMDf)/#100
190             =mEAR   EQU    (mCMDf)+#18  Enable AsynchRonous IDYs
191             =mUNL   EQU    (mCMDf)+#3F  Unaddress listeners
192             =mUNT   EQU    (mCMDf)+#5F  Unaddress talkers
193             =mIFC   EQU    (mCMDf)+#90  InterFaCe clear!!!
194             =mRDYf  EQU    #1500        ReaDY frame
195             =mIDYf  EQU    #1600        IDY frame
196             =mETO   EQU    (mRDYf)+#40  ETO
197             =mETE   EQU    (mRDYf)+#41  ETE
198             *
199             * Single-nibble parameter class
200             *
201             =mADDRM EQU    #2000        ADDRess Me as...
202             =maddrT EQU    #4           ...Talker
203             =maddrL EQU    #2           ...Listener
204             =mUNADM EQU    (mADDRM)+#10 UNADdress Me as...^
205             =mPDLOP EQU    #30          Power down the loop
206             *
207             * Address class
208             *
209             =mADDRT EQU    #4000        ADDRess ... as Talker
210             =mADDRL EQU    #5000        ADDRess ... as Listener
211             =mFINDD EQU    #6000        FIND Device, type n
212             =mFIND1 EQU    (mFINDD)/#1000 FIND Device, type n (1 nibble)
213             =mAUTOA EQU    #70          AUTO Address loop
214             =mAUTOS EQU    (mAUTOA)+1   AUTO Address (AES, AAD)
215             *
216             * Conversation descriptors
217             *
218             =mSDA   EQU    #800000      Start DAta conversation
219             =mSDA@5 EQU    (mSDA)/#100000 Start DAta conversation (P=5)
220             =mSST   EQU    #900000      Start STatus  "
```

```
221             =mSDI   EQU    #A00000       Start Device Id
222             =mSAI   EQU    #B00000       Start Accessory Id
223             =mTCT   EQU    #C00000       Transfer ConTrol
224             =mSETTO EQU    #D00000       SET TimeOut
225             =mSTO@5 EQU    (mSETTO)/#100000 Set TimeOut (P=5)
226             =mSETFC EQU    #E00000       SET Frame Count
227             =mSFC@5 EQU    (mSETFC)/#100000 Set Frame Count @ nibble 5
228             *
229             * One-byte parameter class
230             *
231             =mSETDR EQU    #F30000       SET Device response
232             =mSETA1 EQU    #F30120       SET Accessory ID length (=1)
233             =mSETAI EQU    #F30321       SET Accessory ID value (=3)
234             =mSETS1 EQU    #F30140       SET Status length (=1)
235             =mSETST EQU    #F30041       SET Status value
236             =mSTS@4 EQU    #F3           SET Status value (at nibble 4)
237             =mSETD1 EQU    #F30610       SET Device ID length (=6)
238             =mSETDI EQU    #F30011       SET Device ID value (first byte)
239             =vDEVID EQU    \17PH\        Value of device ID (=HP71)
240             *
241             =mSETTM EQU    #F400         SET Terminator Mode
242             =mSETTC EQU    #F500         SET Terminator Character
243             =mSETIC EQU    #F600         SET # of IDY Timeouts
244             =mSETIT EQU    #F700         SET IDY Timeout (in mS)
245             =mCLRBF EQU    #F8           Clear data buffers (input&output)
246             =mSPTO  EQU    #F900         Set Serial Poll TimeOut
247             =mSETIM EQU    #FA00         Set interrupt mask
248             =mREADI EQU    #FB           Read interrupt cause
249             =mREADC EQU    #FC           Read last device dependent command
250             =CLRTSR EQU    #FD00         ...CLEAR terminate on SRQ mode
251             =SETTSR EQU    (CLRTSR)+1    ...SET terminate on SRQ mode
252             =mPULOP EQU    #FE           Power up the loop
253             =mSPDIS EQU    #FF00         Disable IDY serial poll
254             =mSPEN  EQU    (mSPDIS)+1    Enable IDY serial poll
255             *
256             * Non-parameter messages
257             *
258             =mNOP   EQU    #00           NO oPeration (check for HS)
259             =mRDADR EQU    #01           ReaD ADdRess table
260             =mSTATS EQU    #02           STATuS request to DiaMond
261             =mSTSTC EQU    #0201         Request status, clear service reques
262             =mENDM  EQU    #03           END of Message
263             =mCSRQ  EQU    #04           Clear SRQ on loop
264             =mSSRQ  EQU    #05           Set SRQ on loop
265             =mERSTS EQU    #06           Request ERror STatuS
266             =mAUTOE EQU    #07           Enter AUTO End mode
267             =mMANUL EQU    #08           Go to manual mode
268             =mSCOPE EQU    #0801         Go into MANUAL mode, retransmit
269             =mAUTO  EQU    #09           Go to auto mode
270             =mUPDSC EQU    #0A00         Update System Controller bit(8/0)
271             =mRSTCA EQU    #0B           Reset current address
272             =mGETCA EQU    #0C           Read current address
273             =mINCCA EQU    #0D           Increment current address
274             =mMADDR EQU    #0E           Return "MY" address
275             =mCLRCA EQU    #0F0000       Clear controller status
```

```
276           =mSETCA EQU    #0F01         (Set controller active)
277           =mTAKEC EQU    #0F03         Take control of the loop
278           *                            (2/0: if 2, then use IFC)
279           =mTAKEI EQU    (mTAKEC)~#90  Take control and send IFC frame
280           =mTAKE0 EQU    (mTAKEC)~#10  Take control and send NOP frame
281           *
282           * Diagnostic class
283           *
284           =mRdMem EQU    #F00000       Read memory (add addr, RAM page)
285           =mWrMem EQU    #F10000       Write memory (add value~address)
286           =mTEST  EQU    #F2           Diamond self-test
287           *
288           *--------------------------------------------------------------
289           *
290           * RAM usage...
291           *
292           =SngDev EQU    4             Single device I/O buffer
293           *
294           * IS-xxx:
295           *       nib:   usage:
296           *       ---    ------
297           *       2-0:   If device address known, address, loop # here
298           *              If not known/assigned/iobuffer, FFF
299           *              If assigned, not HPIL, Fxx, xx<>FF
300           *
301           *         3:   If unassigned/not HPIL, F
302           *              If IO buffer for device ID/volume label, 4
303           *              If type specified, loop # + 1 (nib 3: 1,2,3)
304           *              If address specified, 0
305           *              If this assignment has been "OFF"ed, bit 3 is 1
306           *
307           *       6-4:   If type, nib 6: sequence #, nibs 5-4: Acc id
308           *              If address, 6-4: address, loop #
309           *              If IO buffer, 6-4: io buffer #
310           *              If unassigned (NOT "OFF"ed), FFF
311           *              If not HPIL and nib 3=F, not defined
312           *
313           *--------------------------------------------------------------
314           *
315           * Nibble "DSPSET"
316           *
317           =DispOK EQU    11            Display device is set up
318           =Wallby EQU    10            Display device is a Wallaby
319           =Printr EQU    9             Display device is a printer
320           =LoopOK EQU    8             Loop has not died while in disp
321           *
322           *--------------------------------------------------------------
323           *
324           * Nibble "LOOPST" (bits 8 and 9 are cleared when START is called)
325           *
326           =Offed  EQU    11            If set, USER specified OFF IO
327           =Device EQU    10            Last START found device mode
328           *
329           *--------------------------------------------------------------
330           *
```

```
    331            * MBOX^: (3 nibbles)
    332            *       Middle 3 digits of address of last mailbox used (ie if
    333            *         mailbox was at address #20010 then MBOX^ is #001)
    334            *
    335            *---------------------------------------------------------------
    336 00000              END
```

```
=#Timeo  Abs      30 #0001E -    90
=Attn    Abs      12 #0000C -    15
=CLRTSR  Abs   64768 #0FD00 -   250   251
=CkTape  Abs       5 #00005 -    24
=CloseR  Abs       8 #00008 -    49
=DevID   Abs      63 #0003F -    70
=DevTyp  Abs      31 #0001F -    69
=Device  Abs      10 #0000A -   327
=DispOK  Abs      11 #0000B -   317
=DsAddr  Abs       0 #00000 -    79
=DsDevI  Abs       2 #00002 -    81
=DsDevT  Abs       1 #00001 -    80
=DsLoop  Abs       5 #00005 -    84
=DsNull  Abs       4 #00004 -    83
=DsVolL  Abs       3 #00003 -    82
=Format  Abs       5 #00005 -    46
=InpByt  Abs       6 #00006 -    62
=Loop    Abs     159 #0009F -    73
=LoopOK  Abs       8 #00008 -   320
=MaxRec  Abs       7 #00007 -    63
=Null    Abs     127 #0007F -    72
=OUTPIt  Abs       2 #00002 -    97    98
=Offed   Abs      11 #0000B -   326
=PLOTt   Abs       3 #00003 -    98
=PWrite  Abs       6 #00006 -    47
=Positn  Abs       3 #00003 -    59
=Printr  Abs       9 #00009 -   319
=Read    Abs       2 #00002 -    58
=Read0   Abs       0 #00000 -    56
=Read1   Abs       1 #00001 -    57
=Rewind  Abs       7 #00007 -    48
=SETTSR  Abs   64769 #0FD01 -   251
=Seek    Abs       4 #00004 -    45
=SetBP   Abs       3 #00003 -    44
=SngDev  Abs       4 #00004 -   292
=Timout  Abs    2000 #007D0 -    91
=Verify  Abs      11 #0000B -    52
=VolLbl  Abs      95 #0005F -    71
=Wallby  Abs      10 #0000A -   318
=Write   Abs       2 #00002 -    43
=Write0  Abs       0 #00000 -    41
=Write1  Abs       1 #00001 -    42
=XchgL   Abs      10 #0000A -    51
=XchgT   Abs       4 #00004 -    60
=XfrO1L  Abs       9 #00009 -    50
=XfrO1T  Abs       5 #00005 -    61
=ePARSE  Abs       0 #00000 -   129
=ePIL    Abs       2 #00002 -   131
=eTAPE   Abs       1 #00001 -   130
=hs3BYT  Abs       7 #00007 -   168
=hsAWKE  Abs       2 #00002 -   173
=hsERRO  Abs       4 #00004 -   171
=hsLPRQ  Abs       5 #00005 -   170
=hsMANL  Abs       6 #00006 -   169
=hsMGAV  Abs       0 #00000 -   175
```

```
=hsNRD   Abs       1 #00001 -   174
=hsRQSR  Abs       3 #00003 -   172
=mADDRL  Abs   20480 #05000 -   210
=mADDRM  Abs    8192 #02000 -   201   204
=mADDRT  Abs   16384 #04000 -   209
=mAUTO   Abs       9 #00009 -   269
=mAUTOA  Abs     112 #00070 -   213   214
=mAUTOE  Abs       7 #00007 -   266
=mAUTOS  Abs     113 #00071 -   214
=mCLRBF  Abs     248 #000F8 -   245
=mCLRCA  Abs  983040 #F0000 -   275
=mCMD2   Abs      20 #00014 -   189
=mCMD3   Abs     320 #00140 -   188
=mCMDf   Abs    5120 #01400 -   187   188   189   190   191   192   193
=mCSRQ   Abs       4 #00004 -   263
=mDATA2  Abs      16 #00010 -   185
=mDATAf  Abs    4096 #01000 -   184   185
=mEAR    Abs    5144 #01418 -   190
=mENDM   Abs       3 #00003 -   262
=mENDf   Abs    4608 #01200 -   186
=mERSTS  Abs       6 #00006 -   265
=mETE    Abs    5441 #01541 -   197
=mETO    Abs    5440 #01540 -   196
=mFIND1  Abs       6 #00006 -   212
=mFINDD  Abs   24576 #06000 -   211   212
=mFRAME  Abs    4096 #01000 -   183
=mGETCA  Abs      12 #0000C -   272
=mIDYf   Abs    5632 #01600 -   195
=mIFC    Abs    5264 #01490 -   193
=mINCCA  Abs      13 #0000D -   273
=mMADDR  Abs      14 #0000E -   274
=mMANUL  Abs       8 #00008 -   267
=mNOP    Abs       0 #00000 -   258
=mPDLOP  Abs      48 #00030 -   205
=mPULOP  Abs     254 #000FE -   252
=mRDADR  Abs       1 #00001 -   259
=mRDYf   Abs    5376 #01500 -   194   196   197
=mREADC  Abs     252 #000FC -   249
=mREADI  Abs     251 #000FB -   248
=mRSTCA  Abs      11 #0000B -   271
=mRdMem  Abs15728640 #00000 -   284
=mSAI    Abs11534336 #00000 -   222
=mSCOPE  Abs    2049 #00801 -   268
=mSDA    Abs 8388608 #00000 -   218   219
=mSDA@5  Abs       8 #00008 -   219
=mSDI    Abs10485760 #00000 -   221
=mSETAI  Abs15926049 #30321 -   233
=mSETAl  Abs15925536 #30120 -   232
=mSETCA  Abs    3841 #00F01 -   276
=mSETDI  Abs15925265 #30011 -   238
=mSETDR  Abs15925248 #30000 -   231
=mSETDl  Abs15926800 #30610 -   237
=mSETFC  Abs14680064 #00000 -   226   227
=mSETIC  Abs   62976 #0F600 -   243
=mSETIM  Abs   64000 #0FA00 -   247
```

```
=mSETIT  Abs    63232 #0F700 -   244
=mSETST  Abs15925313 #30041 -   235
=mSETSl  Abs15925568 #30140 -   234
=mSETTC  Abs    62720 #0F500 -   242
=mSETTM  Abs    62464 #0F400 -   241
=mSETTO  Abs13631488 #00000 -   224   225
=mSFC@5  Abs       14 #0000E -   227
=mSPDIS  Abs    65280 #0FF00 -   253   254
=mSPEN   Abs    65281 #0FF01 -   254
=mSPTO   Abs    63744 #0F900 -   246
=mSSRQ   Abs        5 #00005 -   264
=mSST    Abs 9437184 #00000 -   220
=mSTATS  Abs        2 #00002 -   260
=mST@5   Abs       13 #0000D -   225
=mST$@4  Abs      243 #000F3 -   236
=mSTSTC  Abs      513 #00201 -   261
=mTAKEC  Abs     3843 #00F03 -   277   279   280
=mTAKEI  Abs   983952 #F0390 -   279
=mTAKEO  Abs   983824 #F0310 -   280
=mTCT    Abs12582912 #00000 -   223
=mTEST   Abs      242 #000F2 -   286
=mUNADM  Abs     8208 #02010 -   204
=mUNL    Abs     5183 #0143F -   191
=mUNT    Abs     5215 #0145F -   192
=mUPDSC  Abs     2560 #00A00 -   270
=mWrMem  Abs15794176 #10000 -   285
=maddrL  Abs        2 #00002 -   203
=maddrT  Abs        4 #00004 -   202
=p3DATA  Abs       15 #0000F -   123
=pACK    Abs        0 #00000 -   108
=pADDR   Abs        4 #00004 -   112
=pCMD    Abs       12 #0000C -   120
=pDATA   Abs       11 #0000B -   119
=pDIAGL  Abs        3 #00003 -   111
=pDIAGR  Abs        2 #00002 -   110
=pEOT    Abs        6 #00006 -   114
=pETE    Abs        9 #00009 -   117
=pHALTD  Abs        7 #00007 -   115
=pIDY    Abs       14 #0000E -   122
=pIFC    Abs        5 #00005 -   113
=pRDY    Abs       13 #0000D -   121
=pSTATE  Abs        1 #00001 -   109
=pTERM   Abs        8 #00008 -   116
=pUTYPE  Abs       10 #0000A -   118
=s3BYTE  Abs        3 #00003 -   159
=sCONTR  Abs        0 #00000 -   153
=sDATAO  Abs        9 #00009 -   144
=sDATAV  Abs        8 #00008 -   145
=sDIAsr  Abs        1 #00001 -    31
=sDevOK  Abs        8 #00008 -    22
=sERROR  Abs        0 #00000 -   162
=sFirst  Abs        0 #00000 -    27
=sINTR   Abs        4 #00004 -   149
=sLISTR  Abs        1 #00001 -   152
=sLOCKD  Abs       11 #0000B -   142
```

```
=sLoop?  Abs        5 #00005 -    25
=sMANUL  Abs        2 #00002 -   160
=sOVERW  Abs        8 #00008 -    21
=sPOLLE  Abs        6 #00006 -   147
=sPRIVT  Abs       11 #0000B -    19
=sRMOTE  Abs       10 #0000A -   143
=sReadd  Abs        4 #00004 -    26
=sSCNTR  Abs        3 #00003 -   150
=sSRQIN  Abs        1 #00001 -   161
=sSTAND  Abs        7 #00007 -   146
=sSTK    Abs        7 #00007 -    23
=sTALKA  Abs        2 #00002 -   151
=sUNCNF  Abs        5 #00005 -   148
=sUNSEC  Abs       10 #0000A -    20
=vDEVID  Abs825708616 #75048 -   239
```

Input Parameters

  Source file name is NZ&SYM::MS

  Listing file name is NZ/SYM:TI:ML::-1

  Object file name is NZ%SYM: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.