

HP-71 FORTH/Assembler ROM

Internal Maintenance Specification - IMS

HP-71 Assembler Internal Design Specification - IDS

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER

CHHU, 2545 W. Camden Place, Santa Ana, CA 92704 USA

CONTENTS BY DOCUMENT

1. FORTH/Assembler ROM IMS - C O N T E N T S 1
   1. Introduction 2
   2. Primitive & Secondary Structure 3
   3. Compiling A New FORTH Word. 5
   4. Vocabulary Structure. 7
   5. Action of FORGET. 8
   6. ROM Dictionary Search Method. 8
   7. Control Structures
      7.1 Control Structure Markers . . . . . .(7-2). 9
   8. Interpreting From A File 11
   9. Contents of FORTHRAM
      9.1 User Variables. . . . . .(9-1). 12
      9.1 RAM Dictionary. . . . . (9-11). 17
      9.2 PAD . . . . . (9-11). 17
      9.3 FORTH Data Stack. . . . . (9-12). 17
      9.4 Terminal Input Buffer & FORTH Return Stack. (9-12). 17
      9.5 Mass Storage Buffers. . . . . (9-13). 18
   10. The FORTH System Variable ACTIVE. 18
   11. POLL HANDLING
      11.1 VER$ POLL. . . . . (11-1). 19
      11.2 FILE TYPE POLL . . . . . (11-1). 19
      11.3 CONFIGURATION POLL . . . . . (11-1). 19
      11.4 MAIN LOOP POLL . . . . . (11-2). 19
      11.5 INSUFFICIENT MEMORY POLL . . . . . (11-3). 20
      11.6 ERROR POLL . . . . . (11-3). 20
   12. Positioning of FORTHRAM FILE. 21
   13. BASIC/FORTH Interface 23
   14. Headerless Words. 24
   15. Misc. FORTH Details
      15.1 FORTH And The ATTN Key And Service Requests(15-1). 27
      15.2 FORTH Usage of CPU Registers . . . . . (15-1). 27
      15.3 ONERR: FORTH's "On Error Goto" . . . . . (15-2). 27

2. FORTH GLOBALS. 28
3. MR%FTO_INITIALIZATION. 32
4. BASIC --> FORTH KEYWORDS 62
5. Jump_Table, 0, 1, %2 char_words. 71
6. FT2: 3 & 4 CHAR WORDS. 116
7. FT1: 5 CHAR WORDS. 173
8. FT4: SIX CHAR WORDS. 215
9. FT5: 7 & 8 CHAR WORDS. 251
10. FT6: 9 10 ETC. CHAR WORDS. 277
11. FT7: PROLOGUES 288

12. HP-71 Assembler IDS FORTH/Assembler ROM. 296
    C O N T E N T S. 310
    1. Overview 296
    2. Memory Layout During Assembly
       2.1 Memory Requirements. . . . . (2-1) 297
    3. Program Flow 298

Note: Numbers in parentheses are HP page numbers as referenced in the IMS or IDS CONTENTS. The first number is the chapter number; the number following the dash is the page number.

4. Comment Conventions
4.1 Word Description . . . . . (4-1) 299
4.2 Algorithm description. . . . . (4-1) 299
4.3 Forth Word . . . . . (4-1) 299
4.4 Stack Description. . . . . (4-2) 299
5. Variables
5.1 User Variables . . . . . (5-1) 300
5.2 Temporary Variables. . . . . (5-1) 300
6. Symbol Table
6.1 Creation of Symbol Table . . . . . (6-1) 301
6.2 Format of Symbol Table Entries . . . . . (6-2) 302
6.3 Important Symbol Table Words . . . . . (6-2) 302
7. Expression Evaluation
7.1 Backus-Naur Form. . . . . (7-1) 302
7.2 Stack Overflow During Evaluation. . . . . (7-1) 302
8. Opcode Lookup & New Opcodes
8.1 Opcode Tables . . . . . (8-1) 303
8.2 Description of an Opcode Table Entry. . . . (8-1) 303
8.3 Hashing function. . . . . (8-3) 304
8.4 Structure of Opcode Groups. . . . . (8-3) 304
8.5 New Opcodes . . . . . (8-4) 305
8.6 oPARRT Routine (STR -- [PTR] F) . . . . . (8-4) 305
8.7 oPRORT ROUTINE (NUM -- [LEN]) . . . . . (8-5) 305
8.8 COMMON ROUTINE TABLE. . . . . (8-5) 305
9. Macro Expansion Psuedo-ops
10. Assembler Aborting & Ending
10.1 SHUTDOWN (NUM --). . . . . .(10-1) 307
10.2 ASSEM.ABORT (NUM STR --). . . . . .(10-1) 307
10.3 CLEANUP (--). . . . . .(10-2) 307
10.4 CLOSEDOWN (--). . . . . .(10-2) 307
11. Out of Memory Condition
11.1 No Room During Initialization . . . . . .(11-1) 308
11.2 Symbol Table Fills. . . . . .(11-1) 308
11.3 At the Start of Pass Two. . . . . .(11-1) 308
11.4 Listing File Fills. . . . . .(11-1) 308
11.5 Call to Basic . . . . . .(11-2) 308
12. Known Bugs
12.1 Word Not Unique . . . . . .(12-1) 309
12.2 Symbol Table Listing Page . . . . . .(12-2) 309

13. AS0:_FORTH_ASSEMBLER_OPCODE_TA 311
14. AS1:_FORTH_ASSEMBLER_VARS. 368
15. AS2:_FORTH_ASSEMBLER_IO. 385
16. AS3:_FORTH_ASSEMBLER_SYT 408
17. AS4:_FORTH_ASSEMBLER_EXPRESSIO 422
18. AS5:_FORTH_ASSEMBLER_OPCODES 438
19. AS6:_FORTH_ASSEMBLER_PSEUDO-OP 464
20. AS7:_FORTH_ASSEMBLER_PARSING 497
21. AS8:_FORTH_ASSEMBLER_MAIN. 508
E N D 523

Note: Numbers in parentheses are HP page numbers as referenced in the IMS or IDS CONTENTS.

P R E F A C E

This Document is released by Hewlett-Packard under a concept of NOMAS or Not Manufacturer Supported. Hewlett-Packard is unable to provide the desired support to print to their standards and provide telephone and letter support of the information contained herein. They provide the information to the user community with the understanding that the recipient will not be an unplanned support burdon. We are happy to have the information, and while it may stimulate additional questions, we believe that information in this form is better than no information at all.

Additional copies of this document may be ordered from the Club of Hewlett-Packard Handheld Users, CHHU, pronounced "chew," at the address below. CHHU is a volunteer, member supported, noncommercial, international, open, group of HP Handheld users dedicated to applying the HP machines of today and making a contribution to the true personal computer of tomorrow. CHHU publishes specialty publications such as this one and provides its members with a monthly (ten issues per year) publication that covers programs and applications articles for all of HP's handhelds. CHHU also sponsors regional Conferences which include proceedings that are made available to the community.

Readers unfamilar with CHHU may request additional information on the Club by sending a large (9" x 12") self addressed envelope with three ounces of postage attached to the address below. If you are interested in back issues of the "CHHU Chronicle" or a list of other publications be sure to specifically request this information.

CHHU
2545 W. Camden Place
Santa Ana, CA 92704 USA
Telephones:
Office - (714) 754-7757 (NOON-4AM)
Phone Bulletin - (714) 754-4557 (24 Hr.)

Any errors you find should be sent to the address above. If you prepare your own contents or other cross references we would appreciate a copy. We can only make improvements if we share our work. CHHU will provide support by answering questions if the phone number above is used. Any comments or suggestions regarding NOMAS publications are solicited.

FORTH/Assembler ROM

Internal Maintenance Specification

**** 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 Primitive & Secondary Structure

3 Compiling A New FORTH Word

4 Vocabulary Structure

5 Action of FORGET

6 ROM Dictionary Search Method

7 Control Structures
7.1 Control Structure Markers 7-2

8 Interpreting From A File

9 Contents of FORTHRAM
9..1 User Variables 9-1
9.1 RAM Dictionary 9-11
9.2 PAD 9-11
9.3 FORTH Data Stack 9-12
9.4 Terminal Input Buffer & FORTH Return Stack . 9-12
9.5 Mass Storage Buffers 9-13

10 The FORTH System Variable ACTIVE

11 Poll Handling
11.1 VER$ POLL 11-1
11.2 FILE TYPE POLL 11-1
11.3 CONFIGURATION POLL 11-1
11.4 MAIN LOOP POLL 11-2
11.5 INSUFFICIENT MEMORY POLL 11-3
11.6 ERROR POLL 11-3

12 Positioning Of FORTHRAM File

13 BASIC/FORTH Interface

14 Headerless Words

15 Misc FORTH Details
15.1 FORTH And The ATTN Key And Service Requests 15-1
15.2 FORTH Usage Of CPU Registers 15-1
15.3 ONERR: FORTH's "On Error Goto" 15-2

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO CONTACT THE MANUFACTURER

FORTH/Assembler ROM IMS

Introduction	CHAPTER 1

The purpose of this document is to explicate details of the implementation HP-71 FORTH/Assembler ROM. It will not explain every detail of the implementation, rather it will seek to highlight areas which are critical for an understanding of our version of FORTH and the Assembler. It is assumed that the person using this document has access to the code listings, the HP-71 IDS Vols. 1, 2 and 3 and the FORTH/Assembler ROM manual.

The code listings and their contents break down as follows (note that the file name appears in the starred block on the cover sheet of each assembly):

MR&GT0 - contains equates defining addresses of user variables; default values of data and return stacks, TIB, mass storage buffers. Contains no executable code.

MR&FT0 - contains poll handlers, error messages, FORTH system initialization, code to create, position and adjust FORTHRAM.

MR&FTB - contains code for BASIC to FORTH words.

MR&FT1 - contains ROM dictionary jump table, 0, 1 and 2 character words.

MR&FT2 - contains 3 and 4 character words.

MR&FT3 - contains 5 character words.

MR&FT4 - contains 6 character words.

MR&FT5 - contains 7 and 8 character words.

MR&FT6 - contains 9, 10, 11 and 12 character words.

MR&FT7 - contains inner loop, secondary prologues, routines to save and restore FORTH pointers.

MR&AS0 - contains opcode lookup tables.

MR&AS1 - contains Assembler variables.

MR&AS2 - contains code to handle source and listing files as well as object code I/O.

MR&AS3 - contains Assembler symbol table manipulation code.

MR&AS4 - contains lexical analysis, and expression evaluation code.

MR&AS5 - contains all opcodes for HP-71 CPU (internally known as SASM--for Saturn Assembler).

MR&AS6 - contains pseudo-ops for control, constants and macro expansion.

MR&AS7 - contains code to do source line parsing.

MR&AS8 - contains body of Assembler.

Primitive & Secondary Structure	CHAPTER 2

FORTH is compiled as a series of addresses (possibly interspersed with data). The inner loop (=NEXT) is perpetually moving through this list, getting the next address and interpreting it as a CFA (code field address) -- which means that it gets the contents of this CFA and sets the program counter to this value. The inner loop, then, does not distinguish primitives from secondaries, i.e. assembly language routines from FORTH words. In both cases the contents of the CFA are put in the program counter.

The CFA of a primitive thus has nothing more than the address of the first line to be executed. Since the first line always in fact follows the CFA, a primitive looks like this:

```
                < header>
=WORD           (WORD+5)    <----- CFA
WORD+5          < code >
                  ....
                  ....
                 RTNCC       return with carry clear
```

If there were only primitives, this would be a waste of 5 nibbles in every word, but it allows primitives and secondaries to appear the same to the inner loop.

The CFA of a SECONDARY always has the address of a prologue. A prologue, generally, prepares the system to deal with the information which follows. For example the DOCOL (do colon) routine resets the instruction pointer (CPU register D0) to the list of addresses following it, after saving the current instruction pointer on the return stack. It effectively nests the system down one level. Secondaries which use DOCOL end with the address of a routine (=SEMI) which will restore the old instruction pointer thus unnesting the sytem one level. Secondaries using DOCOL have the following form:

```
                < header >
=WORD           =DOCOL      <------CFA
```

```
addr
addr
....
....
=SEMI
```

The system provided prologues are:

DOCOL (do colon) is explained above.

DOCON (do constant) pushes the contents of the next 5 nibbles (higher addresses) to the data stack.

DOFCON (do floating point constant) performs a stack lift and puts the contents of the next 16 nibbles (higher addresses) in to the X register.

DOVAR (do variable) pushes the address of the next memory location (higher address) to the data stack.

DOSTR (do string) pushes the address of the character string (located 4 nibbles beyond--higher memory--the prologue) to the data stack and then pushes the current character count of the string (located 2 nibbles beyond the prologue) to the data stack.

DOSTRA (do string array) ensures that the index parameter is > 0 and <= the maximum dimension for the array, (maximum dimension is stored 2 nibbles beyond the prologue) it then takes the maximum length of each string in the array (stored immediately following the prologue) adds 4 to it (2 nibbles for maximum string length and 2 nibbles for current string length) and multiplies this sum by the (index -1). It pushes the address of the desired string (located 4 nibbles beyond the address calculation result) to the data stack and then pushes the current character count of the string (located 2 nibbles beyond the address calculation result) to the data stack.

DOVOC (do vocabulary) fetches the address of the word most recently defined in this vocabulary

(located 4 nibbles beyond the prologue) and stores it in CURRENT. This become the vocabulary in which word searches are started.

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO CONTACT THE MANUFACTURER

Compiling A New FORTH Word	CHAPTER 3

Let's assume that the user has turned on the HP-71 and typed FORTH [ENDLINE]. The display will show "HP-71 FORTH 1A" Now the user types:

: SIXES 6 DUP ; [ENDLINE]

The following describes how FORTH makes a new definition.

When the display shows the FORTH sign-on message "HP-71 FORTH 1A" FORTH is in EXPECT96 waiting for input from the user. After the user types in the above string and hits [ENDLINE] the input received from the user is moved to the Terminal Input Buffer (TIB) along with 2 nulls which signify the end of input. A variable >IN contains a nibble offset into the TIB pointing to the unprocessed input.

TIB: : SIXES 6 DUP ; [2 nulls]

>IN = 0

FORTH is in execute mode (contents of STATE = 0). INTERPRET (via WORD) finds the string ": " (a colon followed by a space) and moves it (and the space) to the end of the dictionary (pointed to by HERE) as a counted string [1 byte length field=1] and the character ":" followed by a space which is not considered part of the string. >IN is updated to 4. The context vocabulary is searched to see if ":" is recognized as an existing word. It is found (by FIND> which returns its execution--TICK--addr) and executed.

Actions Of Colon

1) The current value of the FORTH Data Stack pointer is stored in CSP.

2) The CONTEXT vocabulary is set to be the same as the CURRENT vocabulary.

3) The Name Field Address (NFA) of the last defined word in this vocabulary is compiled into the dictionary as the Link

Field (LF) of SIXES.

4) WORD is called again and it moves the counted string "SIXES " to the end of the dictionary. >IN now is set to 16.

5) The context vocabulary is searched for any previously defined words named "SIXES".

6) If "SIXES" is found to already be in the dictionary AND the WARN flag is set then "SIXES not unique" is displayed.

7) If the length of the name of the new word, "SIXES", exceeds the allowable name length (contents of WIDTH) the new word is truncated to the maximum allowable width.

8) Space is alloted in the dictionary for the name field of the new word.

9) Set the high bit of the length byte in the name field and in the last character of the name ("S" in this example). The fact that the first and last bytes of the name field have their high bit set is used by TRAVERSE to move through the dictionary.

10) Compile into the dictionary a temporary prologue (code field). This primarily saves space for the real code field address which will be added later.

11) Update the contents of CURRENT to point to the Name Field of the new word, SIXES.

12) Set the smudge bit (bit 5) in the Name Field so that if we are redefining an existing word we will find the old copy of the word rather than the new copy during word searches.

13) Set STATE = C0 to indicate compilation mode.

14) Replace the temporary code field in SIXES with DOCOLON, the runtime colon routine.

INTERPRET, now running in compile mode, uses WORD to find the next string "6 ". This is moved to the end of the dictionary as a counted string. The context dictionary is searched to see if it can find a word "6". This search fails and so NUMBER tries to interpret it as a number.

NUMBER successfully converts the ASCII character "6" into the binary 6. The contents of DPL are set to reflect the fact that we found a single length number (as opposed to a double length number or a floating point number).

INTERPRET calls LITERAL which compiles the runtime literal handler into the dictionary (during execution of SIXES this word will take the contents of the next 5 nibbles and push them onto the FORTH Data Stack). It also compiles a 5 nibble representation of a binary 6 into the dictionary. >IN now is set to 20 (decimal).

INTERPRET uses WORD to find the next string, "DUP ". This is moved to the end of the dictionary as a counted string. The context dictionary is searched to see if it can find a word "DUP". This search succeeds (FIND> returns a true flag, the raw length of the word--here 83 hex--and the execution address of DUP. (Raw length is the length byte of the name field address.) Since the word is not immediate (the raw length of the word, 83, is less than the contents of STATE, C0) the execution address is compiled into the dictionary. >IN now is set to 28 (decimal).

INTERPRET uses WORD to find the next string, ";". This is moved to the end of the dictionary as a counted string. The context dictionary is searched to see if it can find a word ";". This search succeeds. The raw length returned by FIND> is C1 hex and, thus, is NOT less than the contents of STATE, C0. The word is, therefore, an immediate word and is executed instead of being compiled.

Actions Of Semi-colon

1) The current value of the FORTH Data Stack pointer is compared to the value stored in CSP (see Actions of Colon). If these values are different an error is generated.

2) The runtime code for semi-colon is compiled into the dictionary.

3) Execute mode is set by setting STATE = 0.

INTERPRET uses WORD to find the next string and it finds a null. This null is moved to the end of the dictionary as a counted string. The context dictionary is searched to see if it can find a word, null. This search succeeds. The raw length returned by FIND> is C0 hex which means that it is an immediate word and is, therefore, executed.

Actions Of Null

1) Null pops off one level of the FORTH Return Stack--throwing away the return address to INTERPRET.

2) It then returns which restores control to the outer loop.

At this point the OK { 0 } message is displayed and then control moves to QUERY which calls EXPECT96 which is where we started.

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO CONTACT THE MANUFACTURER

Vocabulary Structure — CHAPTER 4

First of all, remember that a vocabulary word is itself just another word with a header in the dictionary. When created with the VOCABULARY statement, it is the latest word in the chain of the current vocabulary. When DEFINITIONS is executed, it becomes the current vocabulary. DEFINITIONS simply puts [CONTEXT] into [CURRENT], so our word (let's call it NEW) must first have become the CONTEXT vocabulary. Therefore executing NEW must accomplish this, and in fact that is all there is to the run-time action of NEW. Aside from this, a vocabulary consists of two pointers: one to the latest word defined while it was CURRENT, and another to the previously defined vocabulary word (such as FORTH). So a vocabulary word looks like this:

```
LFA      < link >     link to previous word in vocabulary
                      to which this word itself belongs
NFA      < header >   e.g. "NEW"
CFA      =DOVOC       address of routine to set CONTEXT
         180A
         < last-word link >
         < previous vocabulary link >
```

The 4 nibbles 180A between the runtime code address and the last-word link are a relic of other FORTH systems.

It is important to note that the previous vocabulary mentioned here is strictly the most recent one in the dictionary, whether or not it is in the search order for NEW or not. To understand why this is so, see FORGET (discussed below).

The last-word link is initialized to point to the NFA of the word itself. Therefore when words are added to the chain, they will link back up with the chain to which this word (e.g. NEW) itself belongs.

DOVOC sets CONTEXT to point to the last-word link. The previous-vocabulary link points at the previous-vocubulary-link field of the previous vocabulary, so that one can easily chain back through vocabularies. The

previous-word link of the FORTH vocabulary is zero to mark the end of the chain.

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO CONTACT THE MANUFACTURER

Action of FORGET	CHAPTER 5

HP-71 FORTH implements a "smart" FORGET which allows the multiple vocabularies to be maintained and any word to be forgotten while still using the linear dictionary structure. Of course words cannot simply be pulled out of the middle of this structure without moving other words up -- but this is not possible since addresses (which would have to be updated) are not distinguishable from data within the compiled word. Therefore one either has to waste space in the dictionary which would never be reclaimed, or else forget all words after the word to be forgotten, even if they belong to other vocabularies. We do the latter.

So FORGET must accomplish the following:

1. FIND the word to be forgotten (of course it must be in the current CONTEXT search order).

2. Make sure that the word is neither in the ROM nor protected by the variable FENCE, i.e. that its address is greater than FENCE.

3. See whether it is prior to the CONTEXT vocabulary; if so this vocabulary will be forgotten, so what we do is just to reset CONTEXT to FORTH.

4. Forget all vocabularies which are after the forgotten word. This is done by getting VOC-LINK, which points to the most vocabulary in the dictionary. The vocabularies are linked to each other in a strictly linear way, since our only concern is to chain back through them all until we find one prior to the forgotten word. (This explains the reference made above in discussing vocabulary structure.) We reset VOC-LINK to point to this one.

5. Look at each vocabulary again, starting from [VOC-LINK], to see if its last word (see vocabulary structure for last-word pointer) is after the forgotten word. If so, we chain back through the words to find one which is okay, and reset the las-word pointer in the vocabulary to point to it. Repeat this for every vocabulary, all the way back to FORTH.

6. Reset dictionary pointer (DP) to reclaim space.

ROM Dictionary Search Method	CHAPTER 6

In typical FORTH implementations words are found by a sequential search through the active vocabulary. In seeking to minimize the time spent looking for words HP-71 FORTH takes a different approach.

When seeking to find a word the FORTH system first does a sequential search through the RAM portion of the context vocabulary. This part is like typical FORTH implementations. However when the search fails to find the word in RAM, FORTH searches ROM in a more efficient manner. (Note that all vocabularies eventually chain back to FORTH and that the link field of FORTH is 0 which signifies to the FORTH system that the desired word is not in RAM.)

The ROM dictionary is broken up into 13 separate linked lists. The first list contains all 0 character words (there is only 1--the null word), the second list contains all 1 character words, etc. There are no words in ROM longer than 12 characters, thus the 13 linked lists. At address E0000 is a jump table with 13 entries (5 nibbles each), one for each linked list. The contents of each entry is a pointer to the name field address of the first word in that chain.

The FORTH search routine (FIND› in MR&FT3) uses the length of the string being searched for (length * 5) as an offset into the table at E0000. It picks up the contents from the entry in the table and uses it as the starting point in its ROM search. In this manner when searching for a 3 character word only other 3 character words are examined. No search time is spent looking for words longer than 12 characters.

Control Structures — CHAPTER 7

The FORTH interpreter acts on each word as it is encountered in the input stream, either executing or compiling depending on the value of STATE and the IMMEDIATE-bit in the word header. So it can check the syntax of compile-time words which come in pairs, triples, etc. only by remembering what words have been compiled beforehand. It does this by putting numbers on the data stack when it compiles such a word (these are all control structure words). The numbers are of no significance in themselves, but serve as codes for the matching words to check.

For instance, IF leaves a 2 on the stack, which is expected by THEN. There is no other legitimate way a 2 can appear on the stack by the time THEN is executed, so if it doesn't see a 2, it concludes that there is a syntax error and aborts. If there was no IF but a pernicious user had an immediate word which left 2 on the stack, THEN will be deceived and bad results will follow.

Since some words go searching down through the stack for markers left by their counterparts, there must be a bottom-of-stack marker relative to the word being defined: it is the data stack marker at the time <colon> is interpreted, stored in the variable CSP. This value will also be checked by <semicolon>; if it is not equal to the stack pointer at that time, it is assumed that an unmatched control structure (e.g. IF without a THEN) was entered, and "definition not finished" is issued. Thus if the user leaves something on the stack at compile-time (by doing,say, "[" and something which alters the stack), he will produce the "definition not finished" error, which might be confusing.

A general rule for control structure words is that whatever they leave on the data stack at compile-time MUST COME IN PAIRS. This is necessitated by the function of LEAVE.

Most words need to leave two things on the stack anyway. Consider the operation of IF...THEN. What must be done is to compile a conditional jump at the IF location, followed by the offset to the THEN location. Of course the THEN location is not known when the IF is compiled. So if

leaves, in addition to the code number 2, a pointer to the place where the offset will later be filled in by THEN. (This pointer is easily obtained: it is HERE after the conditional branch has been compiled in.) THEN pops the 2 after checking for it, gets the next item which it assumes to be the proper address, and fills in the offset to THEN at the address left by IF.

Words like CASE which do not leave an address must still leave a pair of things on the stack; it just leaves a pair of 8's (its code).

ELSE can also use a 2 for its code. Since it is optional, its presence must look the same as IF to THEN. And in fact the action of THEN is the same: store the offset to its address at the address on the stack. ELSE compiles an unconditional rather than conditional branch, and must fill in its own address (or rather the offset to it) at the IF location. Therefore it first executes THEN, and then compiles the unconditional branch, leaving the HERE and 2 on the stack. A similar situation obtains between BEGIN, WHILE and REPEAT.

LEAVE occurs between DO and LOOP or +LOOP. An unconditional jump to the LOOP location must be compiled at each LEAVE location; but other control structures may be nested within the loop, and LOOP's first responsibility is to check for a DO. Therefore LEAVE has to: 1) find a DO marker on the stack (put there since the colon was interpreted), and 2) insert its own marker and address beneath the DO-pair on the stack. Thus LOOP can first find a DO and fill in its address at the DO location, then look for LEAVE markers (arbitrarily many) below it and keep filling in its address at those locations. Because all items left on the stack by control structure words come in pairs, it is possible for LEAVE to look through marker/address pairs left by nested words (e.g. IF inside DO...LOOP) until finding the DO marker (or bottom of stack relative to the word being defined).

7.1 Control Structure Markers

Word	Marker	Actions
DO	3	puts an addr and 3 on data stack.

LEAVE	6	searches through data stack until it finds DO marker. Inserts its own marker, 6, and addr on stack before the DO marker and addr.
LOOP	***	expects to find a DO marker, 3, on top of stack. Uses DO addr to compute offset to start of loop; resolves all LEAVE markers and addresses above DO for branching.
+LOOP	***	same as LOOP
IF	2	puts an addr and a 2 on data stack.
ELSE	2	expects to find IF marker, 2, on top of stack. Resolves IF addr for branching and leaves its own addr and marker, 2, on data stack.
THEN	***	expects to find IF or THEN marker, 2, on top of stack. Resolves IF or THEN address for branching.
BEGIN	1	puts an addr and a 1 on data stack.
WHILE	4	puts an addr and a 4 on data stack.
REPEAT	***	expects to find WHILE marker, 4, and WHILE addr, and also BEGIN marker, 1, and BEGIN addr. Resolves addresses for branching.
UNTIL	***	expects to find BEGIN marker, 1, on data stack and resolves BEGIN addr for branching.
CASE	8	leaves 2 markers, 8 , on data stack.
OF	5	expects either pair of 8's left by CASE in which case it drops the 8's and puts an addr and a marker, 5, on stack; or it expects to find a 7 left by ENDOF in which case it leaves the ENDOF

		marker, 7, and addr and puts its addr and marker, 5, on data stack.
ENDOF	7	expects to find an OF marker, 5, on stack. It resovles the OF addr for branching and leaves its marker, 7, and an addr on stack.
ENDCASE	***	expects to find at least 1 ENDOF marker, 7, and an addr. It resolves all ENDOF addresses for branching.

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO CONTACT THE MANUFACTURER

Interpreting From A File	CHAPTER 8

The word LOADF switches input from the keyboard to a LIF1 text file either in RAM or on mass storage. It is possible for a file being LOADFed to LOADF another file, etc. The depth is restricted only by the number of items on the FORTH Return Stack.

The inputs to LOADF are 2 20 bit numbers, the first of which is the length of the character string specifying the file to be LOADFed and the second, the address of this string. As mentioned above, the user can directly access a file on a tape without first storing it in RAM.

The idea of a block has been changed to mean 1 line from a file rather than a 1024 byte sector on a disk. LOADF opens the file and using mainframe code, creates a "file information block" (fib) entry in the fib General Purpose buffer (a system General Purpose buffer). In this fashion it is not necessary for FORTH to retain the string which specifies the file, and further, it is not necessary to worry about whether the file is in RAM or on mass storage. FORTH calls mainframe routines to open, close, and read from the file. These routines, in turn, interface to the HP-IL module if it is present.

When a file is LOADFed the first line of the file is read into the disc buffer least recently used. The fib # provided by the system is stored into the disc buffer and into a FORTH User Variable SCRFIB (screen fib#). The number in SCRFIB describes the active file. The contents of the buffer are interpreted until the null at the end of the line (placed there by EXPECT96) is reached. At that point the FORTH word "WORD", which finds the next word in the input stream to be handled, checks to see if we are loading from a screen. If we are loading from a screen it further checks to see if we have reached the end of the active file. If EOF is false, WORD causes the next line to be read from the file into the same disc buffer and then it finds the next word to handle.

As with other FORTHs it is possible to have nested file LOADFs. LOADF saves the necessary information to return to the file it is currently interpreting when it is told to

load another file.

When a file is being loaded 4 User Variables are of prime importance. When LOADF begins to execute it stores these values on the FORTH Return Stack.

BLK -- When contents of BLK are zero FORTH expects input from the keyboard. When the contents of BLK are non-zero they represent the line number in a file to be interpreted.

>IN -- The contents of this are used as an offset into either the TIB (Terminal Input Buffer) or the currently active mass storage buffer (PREV @).

SCRFIB -- This variable contains the fib# (file information block number) of the active file, or 0 if no file is active, or the fib# of the file that was being interpreted when an error occurred.

LINE# -- This variable contains the line number of the file being interpreted, or the line number of a file that was being interpreted when an error occurred.

Superficially BLK and LINE# seem to contain the same information. However, BLK contains the number of the line we want to interpret, LINE# contains the number of the line which is currently in a mass storage buffer to be interpreted. BLK controls where input is taken from; when an error occurs while interpreting from a file BLK is reset to 0 while LINE# preserves an error tracking trail (along with SCRFIB).

When EOF (end of file) is reached the above 4 variables are restored and execution continues.

There are 3 mass storage buffers. Each is 208 nibbles long. The maximum length of data allowed per line is 96 bytes. If the data is longer than 96 characters, only the first 96 will be copied into the buffer. However, the byte count will reflect the true number of bytes for the current line. The buffer has the following form:

fib No.	line No.	byte count	data (96	2 null
1 byte	5 nibs	2 bytes	bytes max)	bytes

Contents of FORTHRAM — CHAPTER 9

In the following lines the contents of FORTHRAM are explicated. This explanation assumes that FORTHRAM is at its default size--in other words that it has not been grown or shrunk.

9..1 User Variables

Addresses are given in hexadecimal and size in nibbles is given in hexadecimal also. In most cases the appropriate FORTH word will return the address of the variable in question. The item of interest is usually the contents of this address.

Name: DSTKAD Addr: 2FAFD Size: 5

The current value of the FORTH Data Stack is saved here during calls to Main Frame routines which are likely to destroy the contents of the CPU registers used by FORTH (see FORTH CPU Register Usage). The value of the Data Stack is also saved here when FORTH returns control to BASIC. There is no FORTH word which returns this address.

Name: RSTKAD Addr: 2FB02 Size: 5

The current value of the FORTH Return Stack is saved here during calls to Main Frame routines which are likely to destroy the contents of the CPU registers used by FORTH (see FORTH CPU Register Usage). There is no FORTH word which returns this address.

Name: IREG Addr: 2FB07 Size: 5

The current value of the FORTH Instruction pointer is saved here during calls to Main Frame routines which are likely to destroy the contents of the CPU registers used by FORTH (see

FORTH CPU Register Usage). There is no FORTH word which returns this address.

Name: ACTIVE Addr: 2FB0C Size: 5

The contents of this location determines the state of FORTH--whether it is active, whether it is running BASICX, whether it is inactive, etc. (see section on ACTIVE for a complete explanation of the values this variable can contain and their exact meaning). The code which handles the various polls (Error, Configuration, Version, Main Loop) examines the contents of this location to determine how to proceed. There is no FORTH word which returns this address.

Name: oS0 Addr: 2FB11 Size: 5

This location contains the address of the bottom of the FORTH Data Stack. The contents of this location are altered by GROW and SHRINK. The FORTH word S0 returns the contents of this address, and the FORTH word SP0 returns this address.

Name: oR0 Addr: 2FB16 Size: 5

This location contains the address of the bottom of the FORTH Return Stack. The contents of this location are altered by GROW and SHRINK. The FORTH word RP0 returns this address.

Name: oTIB Addr: 2FB1B Size: 5

This location contains the address of the Terminal Input Buffer. The contents of this location are altered by GROW and SHRINK. The FORTH word TIB returns the contents of this address.

Name: oUSE Addr: 2FB20 Size: 5

This location contains the address of the next Mass Storage buffer to be used. The contents of this location are altered by GROW and SHRINK. The FORTH word USE returns this address.

Name: oPREV Addr: 2FB25 Size: 5

This location contains the address of the most recently accessed Mass Storage buffer. The contents of this location are altered by GROW and SHRINK. The FORTH word PREV returns this address.

Name: oFIRST Addr: 2FB2A Size: 5

This location contains the address of the first Mass Storage buffer in FORTHRAM. The contents of this location are altered by GROW and SHRINK. The FORTH word FIRST returns this address.

Name: oLIMIT Addr: 2FB2F Size: 5

This location contains the address of the 1st nibble beyond the last Mass Storage buffer in FORTHRAM. The contents of this location are altered by GROW and SHRINK. The FORTH word LIMIT returns this address.

Name: oVOC-L Addr: 2FB34 Size: 5

This location contains the address of the vocabulary link field in the most recently defined vocabulary (see section on Vocabularies for more details). There is no FORTH word which returns this address.

Name: oRSIZE Addr: 2FB39 Size: 5

This location contains the maximum record size, in bytes, of a line to be read into a Mass Storage buffer (see description of Mass Storage buffers for more details). The default size is 96 (decimal) There is no FORTH word which returns this address.

Name: o#TIB Addr: 2FB3E Size: 5

This location contains the number of characters put into the Terminal Input Buffer by the outer loop. This count is put here by QUERY. The FORTH word #TIB returns this address.

Name: oWIDTH Addr: 2FB43 Size: 5

This location contains the maximum number of characters allowable in a name. The default (and the maximum) is 31 (decimal). The FORTH word WIDTH returns this address.

Name: oWARN Addr: 2FB48 Size: 5

If the contents of this location is non-zero a warning will be issued when a user defines a word that already exists. No warning is given when the contents of this location are 0. The FORTH word WARN returns this address.

Name: oOKFLG Addr: 2FB4D Size: 5

If the contents of this location are zero the "OK { n }" prompt will be given when FORTH is ready to process more input. The FORTH word OKFLG returns this address.

Name: oBLK Addr: 2FB52 Size: 5

If the contents of this location are zero, input is taken from the keyboard. If the contents of this location are non-zero, input is taken from a file and the value at this location is the line number in the file being interpreted. The FORTH word BLK returns this address.

Name: oIN Addr: 2FB57 Size: 5

The contents of this location represent the nibble offset into the Terminal Input Buffer or Mass Storage buffer at which the next word to be processed can be found. The FORTH word >IN returns this address.

Name: oSPAN Addr: 2FB5C Size: 5

The contents of this location represent the number of bytes read during the last execution of EXPECT96. The FORTH word SPAN returns this address.

Name: oSCFIB Addr: 2FB61 Size: 5

When the contents of this location are non-zero they represent the fib# (see file information block in the HP-71 IDS) of the file being interpreted. When an error occurrs during interpretation from a file the fib# associated with that file remains in this location. The FORTH word SCRFIB returns this address.

Name: oCONTX Addr: 2FB66 Size: 5

The contents of this variable point to the name field address of the last word to be defined in the search vocabulary. The FORTH word CONTEXT returns this address.

Name: oCURR Addr: 2FB6B Size: 5

The contents of this variable point to the name field address of the last word to be defined in the vocabulary to which new definitions will be added. The FORTH word CURRENT returns this address.

Name: oSTATE Addr: 2FB70 Size: 5

The contents of this variable determine whether the FORTH system is in execute mode (0) or compile mode (C0). The FORTH word STATE returns this address.

Name: oBASE Addr: 2FB75 Size: 5

The contents of this variable determine the base in which numbers will be interpreted and displayed. The default base is 10 (decimal). Legal base values should be between 2 and 72 (decimal). The FORTH word BASE returns this address.

Name: oDPL Addr: 2FB7A Size: 5

The contents of this variable are used by NUMBER to let INTERPRET know whether it (NUMBER) found an integer, a double integer or a floating point number. There is no FORTH word which returns this address.

Name: oFLD Addr: 2FB7F Size: 5

Used in some FORTH versions to hold a length field reserved for a number during output conversion. This space may be used by the user's application. There is no FORTH word which returns this address.

Name: oCSP Addr: 2FB84 Size: 5

This location is used to hold the current value of the FORTH Data Stack pointer when compilation starts (a ":" was found). When compilation ends (a ";" is found) the current value of the Data Stack must equal the value stored in this location. (See section on Compiling of a FORTH Word for

more details.) There is no FORTH word which returns this address.

Name: oHLD Addr: 2FB89 Size: 5

This location is used when a number is being formatted for display. It contains the address of the last character inserted into the display string. (See section of Number Formatting for more details.) There is no FORTH word which returns this address.

Name: oFENCE Addr: 2FB8E Size: 5

The contents of this location are checked during FORGET. Words which begin at a lower memory address than the value of this location are protected from FORGET. The default setting of this location is 2FCB1 (just after FORTH in the RAM vocabulary). The FORTH word FENCE returns this address.

Name: oDP Addr: 2FB93 Size: 5

The contents of this location are the address of the first free nibble in the RAM dictionary. This location is up-dated whenever space is ALLOTed. The default setting of this location is 2FCB1. The FORTH word HERE returns the contents of this address.

Name: oBSIZE Addr: 2FB98 Size: 5

This location contains the size, in nibbles, of each Mass Storage buffer. The default setting of this location is 208 (decimal). There is no FORTH word which returns this address.

Name: oLINE# Addr: 2FB9D Size: 5

This location contains the line number being interpreted in the currently active file (see SCRFIB). If an error occurs while interpreting from a file this location contains the line number in which the error occurred. The FORTH word LINE# returns this address.

Name: oBASIC Addr: 2FBA2 Size: 5

The contents of this location are used as both a flag and an

address holder. If the contents of this location are zero then FORTH is not executing a FORTH to BASIC word. If however the contents of this location are non-zero they are the address at which the FORTH to BASIC word is to finish. (See the section of FORTH to BASIC and BASIC to FORTH for more details.) There is no FORTH word which returns this address.

Name: oLOOP Addr: 2FBA7 Size: 5

The contents of this location are used to specify the LOOP over which ENTER and OUTPUT statements will function. At present there are no facilities to make use of the multiple loop capability. There is no FORTH word which returns this address.

Name: oSECD Addr: 2FBAC Size: 5

The contents of this location specify the secondary address to be used in ENTER and OUTPUT statements. (See documentation of ENTER and OUTPUT in HP-71 FORTH manual for more details on secondary addressing). The default setting of this location is 0. The FORTH word SECONDARY returns this address.

Name: oPRIME Addr: 2FBB1 Size: 5

The contents of this location is used by ENTER and OUTPUT to determine the primary address of the desired device. The valid range of this variable is 0 through 31. The default setting of this location is 1. The FORTH word PRIMARY returns this address. (See HP-71 FORTH manual for further explanation of PRIMARY.)

Name: oONERR Addr: 2FBB6 Size: 5

The contents of this location are used by the FORTH error poll handler and the internal FORTH ABORT words (different from the ABORT and ABORT" words available to the user). If the contents of this location are zero then errors proceed normally. However if the contents of this location are non-zero it is assumed that the non-zero value is the execution (TICK) address of a user defined ON ERROR routine. (See section on ONERR for more details.)

Name: oERR? Addr: 2FBBB Size: 5

This contents of this location are used in 3 distinct ways by FORTH error poll handlers. First, it signals that an error occurred in FORTH during a FORTHX statement. Second, it signals that an "insufficient memory" error occurred; and third it signals that an error poll was generated by FORTH code rather than Main Frame routines. (See section on poll handling for more details.)

Name: oLASTX Addr: 2FBC0 Size: 10

The contents of this location are used as the LAST X register in the FORTH floating point stack. The FORTH word L returns this address and the FORTH word LASTX returns the contents of this address to the X register.

Name: oX Addr: 2FBD0 Size: 10

The contents of this location are used as the X register in the FORTH floating point stack. The FORTH word X returns this address.

Name: oY Addr: 2FBE0 Size: 10

The contents of this location are used as the Y register in the FORTH floating point stack. The FORTH word Y returns this address.

Name: oZ Addr: 2FBF0 Size: 10

The contents of this location are used as the Z register in the FORTH floating point stack. The FORTH word Z returns this address.

Name: oT Addr: 2FC00 Size: 10

The contents of this location are used as the T register in the FORTH floating point stack. The FORTH word T returns this address.

Name: oFNAME Addr: 2FC10 Size: 10

This location is used to hold an 8 character string used by file manipulation words (CREATEF, ADJUSTF, FINDF). There is no FORTH word which returns this address.

Name: oINTRP Addr: 2FC20 Size: 5

This location holds the execution (TICK) address of the default INTERPRET word. There is no FORTH word which returns this address.

Name: oCREAT Addr: 2FC25 Size: 5

This location holds the execution (TICK) address of the default CREATE word. There is no FORTH word which returns this address.

Name: oNUMB Addr: 2FC2A Size: 5

This location holds the execution (TICK) address of the default NUMBER word. There is no FORTH word which returns this address.

Name: oCOMMA Addr: 2FC2F Size: 5

This location holds the execution (TICK) address of the default comma "," word. There is no FORTH word which returns this address.

Name: oCCOMA Addr: 2FC34 Size: 5

This location holds the execution (TICK) address of the default byte-comma "C," word. There is no FORTH word which returns this address.

Name: oALLOT Addr: 2FC39 Size: 5

This location holds the execution (TICK) address of the default ALLOT word. There is no FORTH word which returns this address.

Name: oUNIQ Addr: 2FC3E Size: 5

This location holds the execution (TICK) address of the word which displays the "XXX not unique" message. There is no FORTH word which returns this address.

Name: oVARID Addr: 2FC43 Size: 5

The Assembler uses this location to save the ID number of the general purpose buffer which contains its symbol table. When this location is non-zero FORTH will treat its contents as the ID of a general purpose buffer and FORTH will preserve this buffer from being purged by Main Frame routines. The user may use this location to preserve 1 buffer. Note that when the Assembler is run it will not preserve the contents of this variable. The FORTH word VARID returns this address.

Name: oPAGES Addr: 2FC48 Size: 5

The contents of this variable specifies the number of print lines per page when the Assembler is sending its output to a device. The minimum setting is 8. The FORTH word PAGESIZE returns this address. Assembler is run it will not preserve the contents of this variable. The FORTH word VARID returns this address.

Name: oLSTNG Addr: 2FC4D Size: 2C

The contents of this location are a string variable, maximum length of 20 (decimal) plus 1 max length byte and 1 current length byte. It is used by the Assembler to store the name of the listing file or device. The FORTH word LISTING returns the current length of the string in this location as well as the address of the 1st byte in the string at this location.

Name: oPARRT Addr: 2FC79 Size: 5

This location is used by the Assembler. When the Assembler is handling an op-code it does not recognize it will check the contents of this location. If non-zero it assumes that the value is the execution address of a user defined word which will handle one or more new op-codes. There is no FORTH word which returns this address.

Name: oPRORT Addr: 2FC7E Size: 5

This location is used by the Assembler in conjunction with oPARRT. If the user has handled some processing for a new op-code the Assembler will expect this variable to contain an execution address of a word which will finish the processing of the new op-code(s). See the Assembler portion of the IMS for more details on this.

Name: oPSTAT Addr: 2FC83 Size: 5

This location is used by the FORTH system to store status information about the condition of the BASIC system when FORTH was entered.

Name: oGRAB Addr: 2FC88 Size: 5

This location is used by the Assembler to hold the current value in oSCFIB (either 0 or the FIB--file information block--number of the file currently being LOADFed). This allows a user to assemble a file in the middle of loading FORTH secondaries.

9.1 RAM Dictionary

Addr: 2FC8D

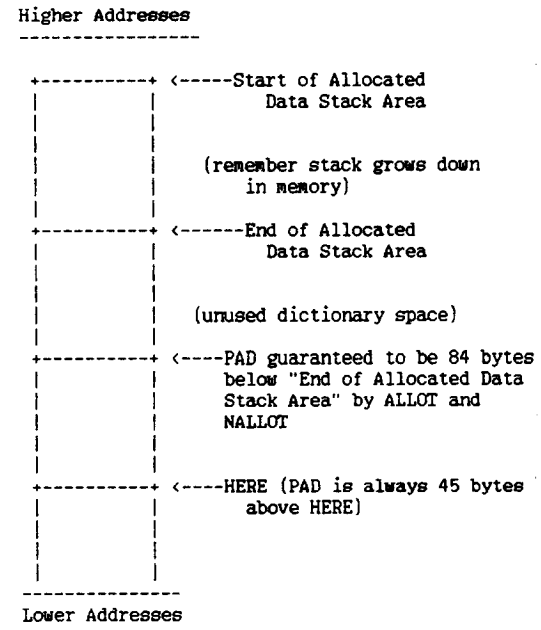
The link field of the word FORTH starts at this address. New words defined by the user are compiled after this word moving toward higher memory addresses.

9.2 PAD

Addr: 2FD0B

The PAD floats above the end of the user RAM dictionary. When space is ALLOTed in the dictionary the address of the PAD will move. Temporary strings and the results of most string operations will be placed in the PAD moving toward high memory. Various algorithms in FORTH seek to ensure that there is always 84 bytes between the start of the PAD area and the top of the Data Stack area.

Numbers being formatted for display (except floating point numbers) are converted into ASCII characters and these characters are temporarily stored directly beneath the PAD. The PAD is situated so that there are 45 bytes between the current end of the dictionary and the beginning of the PAD area.



9.3 FORTH Data Stack

In the default size FORTHRAM file the FORTH Data Stack starts at 30D7C. The stack grows down in memory and there is room for at least 40 entries on the stack. No error is generated if the Data Stack grows into the dictionary area, and, in fact, the system may be corrupted by overwriting the dictionary.

9.4 Terminal Input Buffer & FORTH Return Stack

Two hundred bytes are allocated for both of these structures together. In the default size FORTHRAM the Terminal Input

Buffer begins at 30DC7 (note that this is the bottom of the Data Stack) and is filled toward higher memory addresses. The Return Stack starts at 30F0C and grows down in memory toward the Terminal Input Buffer (TIB).

9.5 Mass Storage Buffers

There are 3 mass storage buffers in the FORTHRAM file. Each is 208 nibbles long. In the default size FORTHRAM the first buffer begins at 30F0C (note that this is the bottom of the Return Stack). See explanation on Mass Storage Buffers in the previous chapter (Interpreting From A File) for more specifics on the mass storage buffers and their usage.

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO CONTACT THE MANUFACTURER

The FORTH System Variable ACTIVE — CHAPTER 10

The location ACTIVE in the file FORTHRAM is used by the system to keep track of where it has been in the many and convoluted paths through BASIC and FORTH. Here are the meaningful values of ACTIVE:

-2	Flag set while handling the error poll, before we send another error poll, so we can ignore this second-order poll.
-1	We went to sleep in FORTH, and have now awakened, realized this in BASPOL, and set the new value in forpon, which will be checked and reset in RESTRT.
0	FORTH has not been called, or it was left with a BYE.
1	Normal condition; FORTH environment is active.
2	BASICI,BASICF or BASIC$ is running; they can't return via BASPOL, so if MainLoop poll occurred we have to go to ABORT.
3	BASICX is running the BASIC environment, ready to return via the MainLoop poll(BASPOL).
[4,5]	There's a check in ERR02 which cares about 2&3 but would also catch 4&5.
99	We just went to sleep by calling power-off in EXPECT96, and have just awakened.

Poll Handling	CHAPTER 11

There are six types of polls handled by FORTH, to gain control back from, or supply needed information to, the mainframe.

11.1 VER$ POLL

When VER$ is executed, the operating system polls the Lex files to get a string identifier from each. We point at the string " FTH:1A " in the ROM, and return saying we handled he poll.

11.2 FILE TYPE POLL

The file FORTHRAM is of a unique type unknown to the mainframe. It issues this poll and we recognize that the file is indeed our type. Then we set up pointers into our own File Type Table, which includes the name of the file-type ("FORTH"), and identifies the file as SECURE if it has been secured. This poll is issued when copying to external media as well as in the CAT function.

11.3 CONFIGURATION POLL

The configuration of the machine can change when the HP-71 wakes up, when a FREEPORT or CLAIMPORT is performed, or when modules are plugged in or pulled out of the machine. If the configuration changes, memory moves. The FORTH ROM intercepts this poll to examine the address of the FORTHRAM file in memory. If FORTHRAM has moved then we must adjust the size of the deadspace within FORTHRAM in order to keep our data at the right address. FORTHRAM must, however, be the first file in the file chain and of the right type,

otherwise we just give up.

If FORTHRAM is the first file in memory and is of the right type we check the contents of oVARID (after ensuring FORTHRAM is in the right place) to see if there is an I/O buffer around which we want to save. This is done by calling a system routine which will re-set (they are first all cleared by the operating system) the high bit in the buffer ID and thus prevent the system from reclaiming

11.4 MAIN LOOP POLL

If FORTH is active, we need to regain control from the operating system. There are several states we might be in, however. We decide what to do based on the value of ACTIVE, as follows:

0 -------- do nothing, RINSXM (FORTH not active)

IF ACTIVE 0:

first check if from FORTHX (oBASIC 0) if so clear ACTIVE, return having handled poll

ELSE

1 -------- assume INIT1 has happened; reset pointers, show sign-on message and ABORT

2 -------- BASIC1/F/$ -- hit a PAUSE (in user function). set ACTIVE to 1 and "WAKEUP" (goto QUIT)

3 -------- BASICX -- collapse HP-71 math stack and restore FORTH pointers; KILL buffer used for BASIC command, clean up items on RTN stack, exit BASICX.

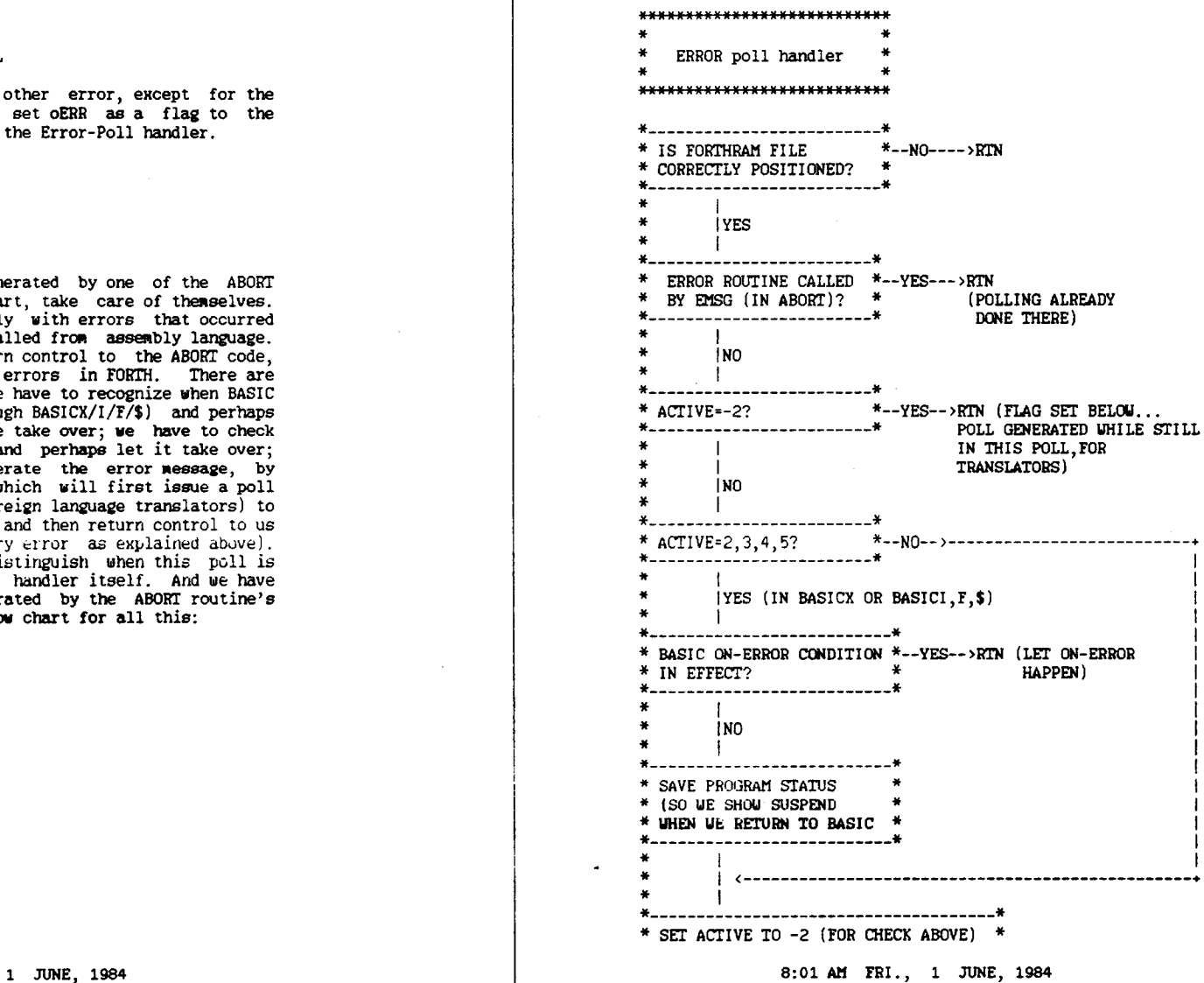
Another way to get here, which is not recognizable from ACTIVE, is through a memory error. In this case a GOSUB to MFERR never returns but goes to MAINLOOP. MEMERR's are flagged by oERR=1 in the error-poll-handler. So when oERR=1 then ABORT.

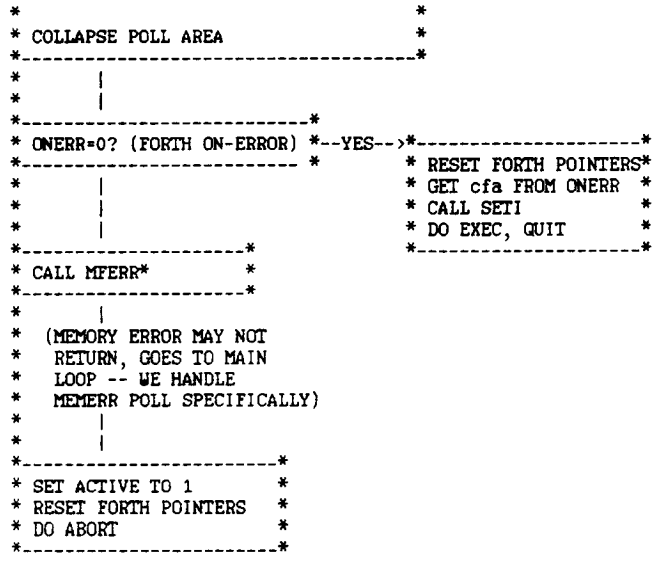
11.5 INSUFFICIENT MEMORY POLL

This is really just like any other error, except for the fact mentioned above. So we set oERR as a flag to the Main-Loop Poll, then jump into the Error-Poll handler.

11.6 ERROR POLL

Errors in FORTH which are generated by one of the ABORT routines will, for the most part, take care of themselves. Here we are concerned primarily with errors that occurred during a mainframe routine called from assembly language. We need to take over and return control to the ABORT code, so as to function like other errors in FORTH. There are many complications, however: we have to recognize when BASIC is running out of FORTH (through BASICX/I/F/$) and perhaps let the BASIC ON-ERROR routine take over; we have to check for the FORTH ONERR condition and perhaps let it take over; and, mainly, we have to generate the error message, by calling the mainframe routine which will first issue a poll to allow other ROM's (like foreign language translators) to substitute their own messages, and then return control to us (except in the case of a memory error as explained above). This means we also have to distinguish when this poll is generated from inside our poll handler itself. And we have to recognize when it is generated by the ABORT routine's primitive, EMSG. Here is a flow chart for all this:





OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER

Positioning Of FORTHRAM File	CHAPTER 12

The functioning of FORTH in the HP-71 is based on absolute addresses. The ROM dictionary is hard configured to always live at E0000. A problem arises when we examine the positioning of FORTH system variables and user definitions. The entire RAM address space is managed by the BASIC O/S; this system works with entities called files. The FORTH system had to respect the integrity of BASIC and therefore all FORTH system variables and user definitions have to exist within the framework of a file. However, files move in RAM and, worse, their movement is unpredictable.

To solve these problems, FORTHRAM is always positioned at the beginning (lowest memory address) of the RAM file chain. Directly beneath the beginning of file memory are the BASIC O/S configuration buffers (see the HP-71 IDS for a complete description of these buffers). These buffers have a minimum and a maximum size. Therefore it is possible to identify the lowest possible start address for a file in the file chain (2FA01) and the highest possible start address (2FAD3). To this latter address we add 25 (hex) nibbles for file overhead (file name, time, creation date, etc.) plus 5 nibbles of storage for FORTH's purposes. This gives us an address at which we can always count on being able to use for FORTH's variables, 2FAFD.

The FORTHRAM file is created, if it does not already exist, when the FORTH or FORTHX keyword is executed. The file is approximately 3K bytes in its default size and enough RAM must exist to create the entire default size file--the file size can be grown or shrunk once FORTH is entered. The difference between the maximum start address and the actual start address is padded with "dead space" (unused RAM) so that FORTH's variables always start at 2FAFD.

Once FORTHRAM exists and is at the beginning of the file chain, the start address of the file will only change when the BASIC O/S re-configures (see HP-71 IDS for conditions when configuration occurs). However the BASIC O/S issues a poll to let ROMs know that configuration has occurred. FORTH sees the poll and adjusts the amount of "dead space" so that its data will still start at 2FAFD.

If FORTHRAM exists but is not at the beginning of the file chain (i.e., FORTHRAM was copied from cassette into memory) when FORTH or FORTHX is executed it will move FORTHRAM to the beginning of the file chain and adjust the amount of dead space to start the data at 2FAFD. There must be at least 37 or more free bytes to affect this change.

The algorithm for creating and/or repositioning FORTHRAM follows:

1) Does FORTHRAM exist? If YES goto 13.

2) Compute the necessary size needed to create FORTHRAM.

   HIGH = (highest possible start addr)

   ACTUAL = actual start addr (contents of BASIC variable MAINST)

   DIFF = HIGH - ACTUAL

3) If DIFF < 0 then memory is in an unpredictable state, error out with Configuration message.

   FSIZET = size needed for FORTH system and user dictionary (5801 nibbles).

   SIZE = FSIZET + DIFF

3) Is there enough memory to create file of SIZE? If YES goto 5.

4) Error out with Insufficient Memory message.

5) Create FORTHRAM file and store address of FORTHRAM in the 5 nibbles immediately following the file header.

6) Initialize FORTH system variables

7) Save BASIS program status.

8) Initialize FORTH Data Stack and Return Stack pointers.

9) Set FORTH as active (ACTIVE = 1). Reset FORTH to BASIC indicator (oBASIC = 0).

10) Did we goto sleep in FORTH and are now waking up? (previous value of ACTIVE = -1)

If so goto 12.

11) Give FORTH sign on message, run QUIT.

12) Get previous value of Data Stack stored in DSTKAD; give wake-up message, run QUIT.

13) Is file correct type? If yes goto 15.

14) Error out with File Type message.

15) Calculate DIFF as above. If DIFF < 0 error out with Configuration message.

16) Is FORTHRAM first file in memory? If not goto 20.

17) Store current address of FORTHRAM immediately following FORTHRAM header. Has FORTHRAM moved in memory since last time we were in FORTH environment (compare old addr stored immediately after FORTHRAM header with current addr of FORTHRAM)? If it has moved goto 19.

18) Reset FORTH ONERR (oONERR=0) and oERR?. Save BASIC program status. Get FORTH Data Stack address from FORTH save area (DSTKAD). Reset FORTH Return Stack pointer. Goto 9.

19) Adjust dead space in FORTHRAM (change file size) to position data at 2FAFD. Goto 18.

20) Are there at least 37 bytes of free memory? If not error out with Insufficient Memory message.

21) Open a hole at the beginning of memory as large as the entire file or as large as available memory allows.

22) Move all or as much as possible of the old FORTHRAM into position as the new FORTHRAM.

23) Shrink old FORTHRAM by amount moved (except for old file's header).

24) Any more to move? (Is link field in header of old FORTHRAM > 5?) If more to move goto 22.

25) Get rid of old FORTHRAM's header. New FORTHRAM

is now in position but its dead space may need
to be adjusted, goto 1.

BASIC/FORTH Interface	CHAPTER 13

FORTH is invoked from the BASIC Interpreter loop via the keywords FORTH or FORTHX. They are statements, meaning that they return no results and are not expected to do anything in particular by the operating system. They end by jumping to the main loop entry point (FORTH) or the Next-statement entry point (FORTHX) -- the latter being almost equivalent to the former except for a few checks such as service requests. FORTHX is programmable and FORTH is not.

The keywords FORTHI/F/$ do not invoke the FORTH system; they simply look into the FORTHRAM file (if it is of the right type and in position) and return values stored there, possibly updating the stack pointer (FORTHI/$). Thus they are not subject to the prohibition against re-entrancy.

This prohibition says that the command string to FORTHX cannot contain BASICX; and that the command string to BASICX cannot contain FORTHX. Also a user-defined function called by BASICI/F/$ cannot call FORTHX. In all these cases it is a question of not saving the environments of the calling function, and not being able to prevent essential pointers (e.g. the continue-address used by Next-statement) from being scrambled. Note however that FORTHX can contain a call to the BASICI/F/$ words. This is because of the different mechanisms used by them and BASICX, as explained below.

BASICX puts the command string in a buffer and jumps to an entry point which is equivalent to executing a statement from the keyboard. In other words we are executing a "program" and the system has to worry about its environment. This becomes apparent when the command statement is "RUN","CALL",etc. The other words, by contrast, are simply passed to the expression- execution routine; they can be thought of as invoking the keyword VAL, whose job is to return a number (or VAL$, which returns a string) to the math stack. Here there is no problem about the program environment, the continue-address and so forth (EXCEPT, let us note, in the irregular and troublesome case of user-defined functions).

Because BASICI/F/$ simply call a function (in effect), they

have control returned to them, and need only pull their result off the math stack. BASICX, on the other hand, has simulated execution from the keyboard and the system will return to the main loop when it is finished. We have to regain control, then, via the main-loop poll.

The BASIC/FORTH interface is complicated by the occurrence of errors. In this case as elsewhere, the value of ACTIVE is used to sort out the appropriate path to take. When an error occurs in BASICX we save away the program status, so that on exit from FORTH we can restore the suspend annunciator, etc. When an error occurs in FORTHX we remember to set the No-continue flag, so a BASIC program will halt after the FORTH error is displayed.

The main idea is to have the two environments continuous within themselves while switching back and forth, in other words whatever is happening in BASIC continues after FORTH has been invoked and exited, and vice versa. For instance, the FORTH data stack is preserved when BASIC becomes active. This result is achieved by maintaining the integrity of FORTHRAM and the ACTIVE variable in it, and by placing the restrictions on re-entrancy mentioned above.

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO CONTACT THE MANUFACTURER

Headerless Words	CHAPTER 14

The following words have no headers and can be accessed only by getting their addresses from the listings. Not included are the runtime actions for immediate words, or vectored words.

MODULE: FT2

name	stack conditions	purpose
DPL	(-- addr)	variable used in determining number of digits to right of decimal point in double integer math
FLD	(-- addr)	variable giving field length in formatting a number for display
CSP	(-- addr)	variable which contains stack addr at start of compilation
HLD	(-- addr)	variable which holds addr of latest char output in numeric display
LIT	(-- n)	pushes contents of next dictionary addr onto stack
PFA	(nfa -- pfa)	convert name field address of compiled word to its parameter field address
FLIT	(--)	pushes the next 16 nibs from the dictionary onto the floating point stack
QCSP	(--)	issue error message if stack position differs from value saved in CSP
PT1+	(--)	advance >IN pointer by one char

MODULE: FT3

F.AND (flag1 flag2 -- flag3)
reverses sense of flag1 and flag2 (i.e., if 0 make it -1, if non-zero make it 0), ORs the result and reverses the sense of the OR. A logical AND as opposed to the provided bitwise AND.

FIND› (addr1 addr2 -- cfa b true)
(-- false)
search through headers to find a dictionary entry; return cfa and first byte of header if found (subroutine of FIND)

VOC-L (-- addr) variable containing addr of the parent-vocabulary field of the most recently created vocabulary

AGAIN
compile-time: (addr 1 -) compiles BRANCH to BEGIN

-FIND (-- cfa b true)
(-- false) calls WORD, gets CONTEXT, and calls FIND›
(subroutine of FIND)

CLEANX resets stack and context, closes all files, flushes disk buffers and unsmudges last dictionary entry (subroutine of ABORT words)

ERRTST execute contents of ONERR if non-zero, else QUIT (subroutine of ABORT words)

ABRT"X (flag --) system version of ABORT" which uses error message table; must be followed by error number in dictionary

ABORTW (flag --) similar to ABRT"X, but prefixes word at HERE to error message, e.g. "FTH ERR: XYZ not recognized"

WHERE (--) display word at HERE

ROM;C (--) stores the runtime address of the

prologue (compiled after it)

BASIC (-- addr) normally zero, variable has addr of return to FORTHX when FORTHX is being executed

MODULE: FT4

CLAP (--) collapse mainframe math stack

?PAIR (n1 n2 --) issue error message if n1 n2

GCHAR (c -- addr t/f) moves string to HERE, looks for ending char c

MOVDP move dictionary pointer past string at end of dictionary

BUFFR (n1 n2 -- addr) obtain nect block buffer, assigning it to block n1 n2=fib of desired file addr=addr of 1st data byte in buffer

M255 (n1 -- n2) if n1›255, n2=255

ROMC+ ROM;C with DOSTR (runtime for string)

ROMC* ROM;C with DOSTRA (runtime for string-array

MODULE: FT5

RSIZE (-- addr) variable with max size of a disk buffer record

ZBRNH (flag --) conditional branch; compile the offset to jump after it in dictionary

BRNCH (--) unconditional branch, followed by offset to jump in dictionary

?NEG (n1 n2 -- n3) if n2‹0 then negate n1

‹;COD ‹;CODE› for RAM defining words (subroutine od DOES›)

FNAME (-- addr) variable with eight character filename set by SYNTAXF

SETNAM	(str --)	set up string with quotes around it and blankfilled, in STMTR0 -- prepares for mainframe routine FSPECx, called by FSYNTAX
CRFIL	(size fnameptr -- filaddr/false)	attempts to create file (subroutine of CREATEF)
NQOT?	(-- flag)	compare next char in stream to quote
D?NEG	(d1 n -- d2)	if n<0, negate d1

MODULE: FT7

DOCOL		pushes I to return stack, makes word-address into new I; found at pfa of all secondaries
DOCON	(-- n)	runtime code for constant (value compiled after it in dictionary) pushes value to stack
DOFCON	(--)	runtime code for floating point constants; does stack-lift and pushes value to X-register
DOVAR	(-- addr)	runtime code for variable; value is after it in word, i.e. at pfa
DOSTR	(-- addr len)	runtime code for string pushes addr of 1st char (=pfa +4) and current length (at pfa+2) to stack
DOSTRA	(n -- addr len)	runtime code for string-array finds nth element and pushes its addr and length
DODOES	(-- pfa)	list of cfa's is compiled after it; it pushes I onto rtn stk; resets I to the cfa list; and pushes pfa of word being executed
DOVOC	(--)	runtime code for a vocabulary word; resets CONTEXT
SAVEFP		saves pointers (CPU registers D1,D0,

	B[A]) in DSTKAD
GETFP	restores pointers saved by SAVEFP
SEMI	pop rtn stack to I (=D0) and go to inner loop; ends every secondary
DOATN	checks to see if ATTN key was hit, and if so runs ABORT; included in BRNCH,ZBRNH and SEMI
GETX	gets contents of X-register as 15-form number in registers (A,B)
GETX+L	calls GETX and puts X in LASTX
PUTABX	puts result of computation (in (A,B)) into X-register
STKLFT	simulates 41-C stack-lift on the "registers" X,Y,Z,T
STKDRP	simulates 41-C stack-drop on the "registers" X,Y,Z,T

Misc FORTH Details	CHAPTER 15

15.1 FORTH And The ATTN Key And Service Requests

The ATTN key functions in 2 ways within FORTH. When the ATTN key is hit while entering input from the keyboard the line is cleared and only the blinking cursor appears. When the ATTN key is hit while a FORTH word is running FORTH executes its internal ABORT routine and again only the blinking cursor is seen.

Hitting the ATTN key, and in fact any key, causes a hardware interrupt. The Main Frame has a vectored routine to handle this key interrupt. It is possible to disable the ATTN key--while a word is executing, not when taking input from the keyboard--by storing some non-zero value in the nibble at 2F441. If the ATTN key has not been disabled this code will set a flag at 2F442.

FORTH checks to see if the ATTN key has been hit each time it executes the runtime semi-colon code (at the end of each secondary) and also each time the runtime branching code (0BRANCH and BRANCH) are executed. If the ATTN key has been hit FORTH clears the ATTN hit flag (2F442) and runs its internal ABORT word. If the ATTN key has not been hit FORTH further checks the exception flag to see if any device has requested service. If the service request flag is set it issues the exception poll (see HP-71 IDS for more detailed information on this poll).

15.2 FORTH Usage Of CPU Registers

D0 : contains FORTH Instruction Pointer

D1 : contains FORTH Data Stack Pointer

B(A) : contains FORTH Return Stack Pointer

Stack Discipline: The stack pointer (Data and Return) is first decremented by 5 BEFORE pushing a new value. The stack pointer is incremented AFTER popping (reading) a value from the stack. Remember that stacks grow down in memory toward lower addresses.

15.3 ONERR: FORTH's "On Error Goto"

When the value contained in oONERR (this address is returned by ONERR) is non-zero, FORTH assumes that it is the execution address of a word the user wishes to run whenever there is a FORTH error.

This location is zeroed out whenever the FORTH environment is entered (FORTH, FORTHX) and also during an INIT1 when FORTH is active.

FORTH often makes use of Main Frame subroutines to accomplish certain tasks. Many of these routines jump immediately to the Main Frame error handler if they encounter a problem without first giving FORTH a chance to handle the error. This fact makes the usage of ONERR problematic but still a useful tool. Before FORTH calls a Main Frame routine which might error out it saves away the FORTH system pointers (see section of FORTH Usage Of CPU Registers). In these cases FORTH gains control of the machine when the BASIC O/S issues the error poll. FORTH sees that ONERR is set and, after restoring the FORTH system pointers from the save area, runs the user's ON ERROR routine. Neither the FORTH Return or Data stacks are reset. It is the responsiblity of the user to determine what useful information is on the stacks for a particular error.

If the user's routine simply returns without cleaning up the stacks, FORTH will run the QUIT word (otherwise the system would crash). The user may attempt to handle the problem and pop the appropriate number of addresses off the FORTH Return Stack to return to a point in their application where execution may safely continue.

```
                TITLE   FORTH_GLOBALS_<840608.1403>

*
* FORTH RAM File:   Addrs are approximate at this point.
*
*       File Size=5969 nibbles max (approx 3K bytes)
*
*       +-------------------------------------+
*       | File Overhead (37 nibbles)          |
*       | RAM file address (5 nibbles)        |
*       |                                     |
*       | --floating area (210 nibbles Max)-- |
*       |                                     |
*       | #2FAFD  Housekeeping                |
*       |                                     |
*       |      DATA STACK SAVE AREA           |
*       |      RTN STACK SAVE AREA            |
*       |      IREG SAVE AREA                 |
*       |                                     |
*       | #2FB11 Start of USER VARIABLE TABLE |
*       |                                     |
*       | #2FCB1 Dictionary Start             |
*       |                                     |
*       |  PAD                                |
*       |                                     |
*       |  Text Buffer: temporary holding     |
*       |  area for strings                   |
*       |-------------------------------------|
*       |     /|\                            |
*       |      |                              |
*       | DATA STK Start                      |
*       |-------------------------------------|
*       |            Terminal Input Buffer    |
*       |                      |              |
*       |     /|\             \|/             |
*       |      |                              |
*       | RTN STK Start                       |
*       |-------------------------------------|
*       | Disk Buffer Area (612 nibbles)      |
*       | ( 3 102 byte buffers)               |
*       +-------------------------------------+
*
```



```
                EJECT

*
* FILE EQUATES: see discussion on how DSTKAD is obtained
*               at end of this file
*

=DSTKAD  EQU   #2FAFD        DATA STK Save area
=RSTKAD  EQU   (DSTKAD)+5    RTN STK Save area
=IREG    EQU   (RSTKAD)+5    "I" REG Save area
=ACTIVE  EQU   (IREG)+5      Says if FORTH is active or not
* ----------------------------
=UVARP   EQU   20            SUM OF ITEMS BEFORE UVAR
=UVAR    EQU   (ACTIVE)+5    Start of User Variable Table

*-------------------------------------------------------
*
* U S E R   V A R I A B L E S
*
*-------------------------------------------------------


=oS0     EQU  (UVAR)+0       Data Stack Ptr
=oR0     EQU  (UVAR)+5       Rtn Stack Ptr
=oTIB    EQU  (UVAR)+10     TERMINAL INPUT BUFFER POINTER
=oUSE    EQU  (UVAR)+15     NEXT BUFFER TO USE
=oPREV   EQU  (UVAR)+20     MOST RECENT DISK BUFFER
=oFIRST  EQU  (UVAR)+25     ADDR OF 1ST DISC BUFFER
=oLIMIT  EQU  (UVAR)+30     ADDR OF 1ST NIB BEYOND BUFFERS
=oVOC-L  EQU  (UVAR)+35     VOCABULARY LINK
=oRSIZE  EQU  (UVAR)+40     RECORD SIZE IN BYTES OF BUFFER
=o#TIB   EQU  (UVAR)+45     # CHARS IN TIB, PUT THERE BY QUERY
=oWIDTH  EQU  (UVAR)+50     MAX NAME LENGTH (DEFAULT=31)
=oWARN   EQU  (UVAR)+55     WARNING MODE
=oOKFLG  EQU  (UVAR)+60     ENABLE/DISABLE "OK" IN QUIT
=oBLK    EQU  (UVAR)+65     CURRENT DISK BLOCK BEING LOADED (0=TE
=oIN     EQU  (UVAR)+70     OFFSET IN TERMINAL INPUT BUFFER
=oSPAN   EQU  (UVAR)+75     # CHARS READ DURING LAST EXPECT96
=oSCFIB  EQU  (UVAR)+80     CURRENT FORTH DISK SCREEN
=oCONTX  EQU  (UVAR)+85     CONTEXT
=oCURR   EQU  (UVAR)+90     CURRENT
=oSTATE  EQU  (UVAR)+95
=oBASE   EQU  (UVAR)+100
=oDPL    EQU  (UVAR)+105    OFFSET TO DEC PT AFTER DINT INPUT
=oFLD    EQU  (UVAR)+110    OUTPUT FIELD WIDTH
=oCSP    EQU  (UVAR)+115    CURR STK POS (USED BY COMPILER)
=oHLD    EQU  (UVAR)+120    POINTS TO LAST CHAR IN PAD
=oFENCE  EQU  (UVAR)+125    PREVENT FORGETTING BELOW FENCE
=oDP     EQU  (UVAR)+130    DICTIONARY POINTER=NEXT AVAIL NIB
=oBSIZE  EQU  (UVAR)+135    SIZE OF BUFFERS IN NIBBLES
=oLINE#  EQU  (UVAR)+140    CURRENT LINE IN ACTIVE BUFFER
=oBASIC  EQU  (UVAR)+145    RETURN ADDR FOR BASIC KEYWORDS
=oLOOP   EQU  (UVAR)+150    selected HPIL LOOP (0,1 or 2)
=oSECD   EQU  (UVAR)+155    SECONDARY loop addr, if any
=oPRIME  EQU  (UVAR)+160    PRIME loop address
```

```
 98          =oONERR  EQU  (UVAR)+165    ON ERROR flag
 99          =oERR?   EQU  (UVAR)+170    ERROR OCCURRED?
100
101          *
102          *  (more user variables below)
103          *---------------------------------------------------------
104
105          =BUFLEN  EQU  208           208 NIBS LONG
106          =RECSIZ  EQU  96            RECORD IS 96 BYTES LONG
107
108          *---------------------------------------------------------
109          * X, Y, Z, T AND L FLOATING POINT STACK
110          *---------------------------------------------------------
111
112          =oLASTX EQU  (UVAR)+175      LAST X register
113          =oX     EQU  (UVAR)+191      X register
114          =oY     EQU  (UVAR)+207      Y register
115          =oZ     EQU  (UVAR)+223      Z register
116          =oT     EQU  (UVAR)+239      T register
117
118          *---------------------------------------------------------
119          =oFNAME  EQU  (UVAR)+255    FILENAME SET BY FSYNTX,USED BY CRFILE
120          *                           EIGHT BYTES LONG
121          *---------------------------------------------------------
122
123          *
124          * LOCATIONS FOR INDIRECT EXECUTION OF FOLLOWING 6 WORDS
125          *
126
127          =oINTRP  EQU  (UVAR)+271    INTERPRET
128          =oCREAT  EQU  (UVAR)+276    CREATE
129          =oNUMB   EQU  (UVAR)+281    NUMBER
130          =oCOMMA  EQU  (UVAR)+286    COMMA ,
131          =oCCOMA  EQU  (UVAR)+291    C-COMMA C,
132          =oALLOT  EQU  (UVAR)+296    ALLOT
133          =oUNIQ   EQU  (UVAR)+301    FOR "ISN'T UNIQUE MESSAGE"
134          *
135          * ASSEMBLER VARIABLES
136          *
137
138          =oVARID  EQU  (UVAR)+306    ASSEMBLER VARIABLE
139          =oPAGES  EQU  (UVAR)+311    "           "
140          =oLSTNG  EQU  (UVAR)+316    ASSEMBLER STRING VARIABLE
141          *                           44 NIBBLES LONG
142          =oPARRT  EQU  (UVAR)+360
143          =oPRORT  EQU  (UVAR)+365
144          *
145          * STILL MORE VARIABLES
146          *
147          =oPSTAT  EQU   (UVAR)+370   PROGRAM STATUS STUFF
148          =oGRAB   EQU   (UVAR)+375   used by ASSEMBLER to hold SCRFIB#
149          *
150          =TSIZE   EQU  380
151          =ENDVAR  EQU  (UVAR)+(TSIZE)
152          *
```



```
153          * FRTHHD MUST BE SET TO (ENDVAR)+5 INSTEAD OF (ENDVAR) BECAUSE
154          * IT IS PRECEEDED BY 5 NIBS OF 0 SIGNIFYING NO MORE LINKS IN RAM
155          *
156          =FRTHHD  EQU  (ENDVAR)+5      RAM DICTIONARY ENTRY FOR FORTH
157          *
158
159          =FRTHVC  EQU  (FRTHHD)+#15    ( ' FORTH 5 + ) CURRENT/CONTEXT fie
160          =FRTHVL  EQU  (FRTHHD)+#1A    ( ' FORTH 7 + ) VOC-LINK field
161          =DICTST  EQU  (FRTHHD)+#1F    start of available dictionary space
162          =TBLLEN  EQU (UVARP)+(TSIZE)+5+#24  STARTING STUFF FROM DSTKAD TH
163          *                             UVARS+FRTHHDLINK+FRTHVC,ETC.
164          *---------------------------------------------------------
165          * ERROR MESSAGE NUMBERS
166          *---------------------------------------------------------
167          =eRROR  EQU      0          GOES EVERYWHERE
168          =eNEND  EQU      1          no ending something
169          =eNOTF  EQU      2          xxx Not Found
170          =eNO;   EQU      3          no ending ;
171          =eNOEP  EQU      4          no ending )
172          =eNO"   EQU      5          no ending "
173          =eLT1   EQU      6          less than 1
174          =eNDEF  EQU      7          def not finished
175          =eDFUL  EQU      8          dictionary full
176          =eCOMP  EQU      9          compile only
177          =ePILE  EQU     10          HPIL error
178          =eRDNL  EQU     11          redefined null
179          =eBADD  EQU     12          bad dictionary entry
180          =ePDIC  EQU     13          in protected dictionary
181          =eEMPT  EQU     14          empty stack
182          =eFULL  EQU     15          full  stack
183          =eNREC  EQU     16          not recognized
184          =eCOND  EQU     17          conditionals not paired
185          =eNOFTH EQU     18          wrong # of items left on stack
186          =eBFIL  EQU     19          bad file specifier
187          *
188          *
189          *
190          =eRANGE EQU     43          address out of range
191          *
192          =eDUPFL EQU     45          FILE EXISTS
193          =eBADPM EQU     46          BAD PARAMETERS
194          *
195          =eCONF  EQU     52          CONFIGURATION
196          =eSWF   EQU     53           STRING WON'T FIT
197          =eNCUR  EQU     54          NOT IN CURRENT VOCABULARY
198          =eCNLD  EQU     55            CANNOT LOAD
199          =eNODO  EQU     59           NO DO BEFORE LEAVE
200          =eCASE  EQU     60           BAD CASE STATEMENT
201          * JEFF
202          =eREENT EQU     62         not re-entrant
203          =eBASCX EQU     63          BASIC not re-entrant
204          =eFORTX EQU     64          FORTH not re-entrant
205          *---------------------------------------------------------
206          *More Useful Equates
207
```

```
208           *CELL    EQU  5           5 nibbles
209
210           *NAMLEN  EQU  31          word name length *31
211
212           *FSIZE   EQU  5759        REQUIRED START SIZE OF FORTH FILE:
213           *
214           *                               20 NIBS HOUSEKEEPING
215           *                              275 NIBS USER VARIABLE TABLE
216           *                               80 NIBS X,Y,Z,T & L STACK
217           *                             4096 NIBS FOR DICTIONARY (2K BYTES)
218           *                               64 NIBS FOR PAD
219           *                              200 NIBS DATA STACK (40 ENTRIES)
220           *                              400 NIBS TIB & RTN STACK
221           *                              612 NIBS 3 102 BYTE DISC BUFFS
222           *                             ----
223           *                             5759 NIBS
224
225
226           *MINFIL  EQU  (DSTKAD)-37-5     ADJUSTABLE STARTING ADDR
227
228
229
230
231           *FSIZET  EQU  (FSIZE)+37+5
232           *
233           *                               37 NIBS FILE HEADER
234           *                                5 NIBS RAM FILE START ADDR
235           *                             5759 NIBS FOR FIXED FORTH FILE
236           *
237
238           *FILEST EQU  #2FA01          This is the theoretical lowest
239           *                           starting address for the 1st file
240           *                           in the file chain.  The actual
241           *                           start addr (found in MAINST) will
242           *                           determine the amount of dead space
243           *                           required in the FORTHRAM file.
244           *
245           *
246           *  The lowest addr. is #2FA01 plus the max. size of the
247           *  configuration buffer gives us the highest possible
248           *  start addr of #2FAD3.  This would be the start of the file
249           *  to which we would add the 37 nibs of file overhead and
250           *  5 nibs to hold the RAM addr of our file the last time
251           *  we were in FORTH gives us the addr at which we can always
252           *  count on being able to start our data at:  #2FAFD
253           *
254
255           *TOP     EQU  ((FSIZE)+(DSTKAD))
256
257           *DISKBF  EQU  624         ALLOW 3 104 BYTE BUFFERS
258
259           *TERMBF  EQU  400         200 bytes for RTN STK & Terminal
260           *                          input buffer
261
262           *PADOFF  EQU  64          Offset from top of dictionary
```



```
263           *                           to start of Text Buffer. 64 NIBS
264           *                           allows 31 chars and 1 length byte
265           *                           which is the max name length.
266
267           *RSTK    EQU  ((TOP)-(DISKBF))   Start of RTN STK
268
269           *USEIT   EQU  ((RSTK)+(BUFLEN))  BUFFER TO USE FIRST
270
271           *LAST+1  EQU  (TOP)              BUFFER LIMIT
272
273           *TIBF    EQU  ((RSTK)-(TERMBF))  Terminal Buffer Start
274
275           *DSTK    EQU  (TIBF)             Start of DATA STK
276
277           *TXBUF   EQU  ((DICTST)+(PADOFF))  Start of Text Buffer
278
279           *FthFil  EQU  #E218       FORTH RAM FILE TYPE
280           *F7STRT  EQU  #E7000       BEGINNING OF 7TH MODULE
281           *TYTABF  EQU  F7STRT      TYPE TABLE, REFERENCED IN FTO,
282           *                         IS FIRST THING IN FT7
283           *ID      EQU  47          IDENTIFICATION # FOR OUR ROM
284           *
285           * TABLE OF CONSTANTS LEFT BY CONTROL STRUCTURE WORDS
286           * FOR SYNTAX CHECKING AT COMPILE TIME
287           *
288           * word leaving
289           * value on stack              word checking for it
290           * BEGIN          1            UNTIL
291           * IF             2            THEN
292           * DO             3            LOOP,+LOOP
293           * WHILE          4            REPEAT
294           * OF             5            ENDOF
295           * LEAVE          6            LOOP,+LOOP
296           * ENDOF          7            ENDCASE
297           * CASE           8            OF
298
299
300 00000                 END
```

=ACTIVE	Abs	195340	#2FB0C -	53	56						
=BUFLEN	Abs	208	#000D0 -	105	269						
=CELL	Abs	5	#00005 -	208							
=DICTST	Abs	195761	#2FCB1 -	161	277						
=DISKBF	Abs	624	#00270 -	257	267						
=DSTK	Abs	200060	#30D7C -	275							
=DSTKAD	Abs	195325	#2FAFD -	50	51	226	255				
=ENDVAR	Abs	195725	#2FC8D -	151	156						
=F7STRT	Abs	946176	#E7000 -	280	281						
=FILEST	Abs	195073	#2FA01 -	238							
=FRTHHD	Abs	195730	#2FC92 -	156	159	160	161				
=FRTHVC	Abs	195751	#2FCA7 -	159							
=FRTHVL	Abs	195756	#2FCAC -	160							
=FSIZE	Abs	5759	#0167F -	212	231	255					
=FSIZET	Abs	5801	#016A9 -	231							
=FthFil	Abs	57880	#0E218 -	279							
=ID	Abs	47	#0002F -	283							
=IREG	Abs	195335	#2FB07 -	52	53						
=LAST+1	Abs	201084	#3117C -	271							
=MINFIL	Abs	195283	#2FAD3 -	226							
=NAMLEN	Abs	31	#0001F -	210							
=PADOFF	Abs	64	#00040 -	262	277						
=RECSIZ	Abs	96	#00060 -	106							
=RSTK	Abs	200460	#30F0C -	267	269	273					
=RSTKAD	Abs	195330	#2FB02 -	51	52						
=TBLLEN	Abs	441	#001B9 -	162							
=TERMBF	Abs	400	#00190 -	259	273						
=TIBF	Abs	200060	#30D7C -	273	275						
=TOP	Abs	201084	#3117C -	255	267	271					
=TSIZE	Abs	380	#0017C -	150	151	162					
=TXBUF	Abs	195825	#2FCF1 -	277							
=TYTABF	Abs	946176	#E7000 -	281							
=USEIT	Abs	200668	#30FDC -	269							
=UVAR	Abs	195345	#2FB11 -	56	65	66	67	68	69	70	71
				72	73	74	75	76	77	78	79
				80	81	82	83	84	85	86	87
				88	89	90	91	92	93	94	95
				96	97	98	99	112	113	114	115
				116	119	127	128	129	130	131	132
				133	138	139	140	142	143	147	148
				151							
=UVARP	Abs	20	#00014 -	55	162						
=eBADD	Abs	12	#0000C -	179							
=eBADPM	Abs	46	#0002E -	193							
=eBASCX	Abs	63	#0003F -	203							
=eBFIL	Abs	19	#00013 -	186							
=eCASE	Abs	60	#0003C -	200							
=eCNLD	Abs	55	#00037 -	198							
=eCOMP	Abs	9	#00009 -	176							
=eCOND	Abs	17	#00011 -	184							
=eCONF	Abs	52	#00034 -	195							
=eDFUL	Abs	8	#00008 -	175							
=eDUPFL	Abs	45	#0002D -	192							
=eEMPT	Abs	14	#0000E -	181							
=eFORTX	Abs	64	#00040 -	204							

=eFULL	Abs	15	#0000F -	182
=eLT1	Abs	6	#00006 -	173
=eNCUR	Abs	54	#00036 -	197
=eNDEF	Abs	7	#00007 -	174
=eNEND	Abs	1	#00001 -	168
=eNO"	Abs	5	#00005 -	172
=eNO;	Abs	3	#00003 -	170
=eNODO	Abs	59	#0003B -	199
=eNOEP	Abs	4	#00004 -	171
=eNOFTH	Abs	18	#00012 -	185
=eNOTF	Abs	2	#00002 -	169
=eNREC	Abs	16	#00010 -	183
=ePDIC	Abs	13	#0000D -	180
=ePILE	Abs	10	#0000A -	177
=eRANGE	Abs	43	#0002B -	190
=eRDNL	Abs	11	#0000B -	178
=eREENT	Abs	62	#0003E -	202
=eRROR	Abs	0	#00000 -	167
=eSWF	Abs	53	#00035 -	196
=o#TIB	Abs	195390	#2FB3E -	74
=oALLOT	Abs	195641	#2FC39 -	132
=oBASE	Abs	195445	#2FB75 -	85
=oBASIC	Abs	195490	#2FBA2 -	94
=oBLK	Abs	195410	#2FB52 -	78
=oBSIZE	Abs	195480	#2FB98 -	92
=oCCOMA	Abs	195636	#2FC34 -	131
=oCOMMA	Abs	195631	#2FC2F -	130
=oCONTX	Abs	195430	#2FB66 -	82
=oCREAT	Abs	195621	#2FC25 -	128
=oCSP	Abs	195460	#2FB84 -	88
=oCURR	Abs	195435	#2FB6B -	83
=oDP	Abs	195475	#2FB93 -	91
=oDPL	Abs	195450	#2FB7A -	86
=oERR?	Abs	195515	#2FBBB -	99
=oFENCE	Abs	195470	#2FB8E -	90
=oFIRST	Abs	195370	#2FB2A -	70
=oFLD	Abs	195455	#2FB7F -	87
=oFNAME	Abs	195600	#2FC10 -	119
=oGRAB	Abs	195720	#2FC88 -	148
=oHLD	Abs	195465	#2FB89 -	89
=oIN	Abs	195415	#2FB57 -	79
=oINTRP	Abs	195616	#2FC20 -	127
=oLASTX	Abs	195520	#2FBC0 -	112
=oLIMIT	Abs	195375	#2FB2F -	71
=oLINE#	Abs	195485	#2FB9D -	93
=oLOOP	Abs	195495	#2FBA7 -	95
=oLSTNG	Abs	195661	#2FC4D -	140
=oNUMB	Abs	195626	#2FC2A -	129
=oOKFLG	Abs	195405	#2FB4D -	77
=oONERR	Abs	195510	#2FBB6 -	98
=oPAGES	Abs	195656	#2FC48 -	139
=oPARRT	Abs	195705	#2FC79 -	142
=oPREV	Abs	195365	#2FB25 -	69
=oPRIME	Abs	195505	#2FBB1 -	97
=oPRORT	Abs	195710	#2FC7E -	143

```
aturn Assembler     FORTH_GLOBALS_<840109.1042>      Mon Jan  9, 1984  10:42 am
er. 3.33/Rev. 2241  Symbol Table                                      Page  9

oPSTAT  Abs  195715 #2FC83 -   147
oR0     Abs  195350 #2FB16 -    66
oRSIZE  Abs  195385 #2FB39 -    73
oS0     Abs  195345 #2FB11 -    65
oSCFIB  Abs  195425 #2FB61 -    81
oSECD   Abs  195500 #2FBAC -    96
oSPAN   Abs  195420 #2FB5C -    80
oSTATE  Abs  195440 #2FB70 -    84
oT      Abs  195584 #2FC00 -   116
oTIB    Abs  195355 #2FB1B -    67
oUNIQ   Abs  195646 #2FC3E -   133
oUSE    Abs  195360 #2FB20 -    68
oVARID  Abs  195651 #2FC43 -   138
oVOC-L  Abs  195380 #2FB34 -    72
oWARN   Abs  195400 #2FB48 -    76
oWIDTH  Abs  195395 #2FB43 -    75
oX      Abs  195536 #2FBD0 -   113
oY      Abs  195552 #2FBE0 -   114
oZ      Abs  195568 #2FBF0 -   115
```

```
Saturn Assembler     FORTH_GLOBALS_<840109.1042>      Mon Jan  9, 1984  10:42 am
Ver. 3.33/Rev. 2241  Statistics                                       Page 10

Input Parameters

  Source file name is MR&GTO

  Listing file name is MR/GTO::65

  Object file name is MR%GTO::65

                                      111111
                              0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News
```

```
Saturn Assembler     MR%FT0_INITIALIZATION__<840206  Mon Feb  6, 1984  11:24 am
Ver. 3.33/Rev. 2241                                                   Page  1

     1                    TITLE MR%FT0_INITIALIZATION__<840608.1403>
     2                    RDSYMB MR%GT0
     3               ************************
     4               *                      *
     5               * File: MR%FR0        *
     6               *                      *
     7               * Re-released 2/6/84   *
     8               *                      *
     9               ************************
    10
    11
    12
    13               *  File Header
    14
    15
    16 00000 0000 =FR0ST  CON(8) 0
             0000
    17 00008 64F4         NIBASC \FORTHROM\      File Name
             2545
             8425
             F4D4
    18
    19 00018 0000         CON(4) (=fLEX)         File Type: LEX
    20 0001C 20           NIBHEX 20              Flags (PRIVATE)
    21 0001E 2541         NIBHEX 2541            Time
    22 00022 9110         NIBHEX 911038          Date
             38
    23 00028 0000         REL(5)  =FILEND         File Length
             0
    24               *
    25 0002D F2           CON(2)  =ID            Id
    26 0002F 10           CON(2)   1             Lowest Token
    27 00031 50           CON(2)   5             Highest Token
    28 00033 0000         NIBHEX 00000           End of lex table chain
             0
    29               *
    30 00038 F            NIBHEX F               Speed table omitted
    31 00039 B300         CON(4) (TxTbSt)+1-(*)  Offset to text table
    32 0003D 2800         CON(4) (MsgTbl)-*      Offset to message table
    33 00041 D8B0         REL(5) POLHND          Offset to poll handler
             0
```

```
                    STITLE M a i n   T a b l e
               *  Main Table
00046          =xromF0
               *
00046 C30      CON(3) 60              01  Invoke FORTH system
00049 C9F0 =xfrth REL(5) =XFORTH
      0
0004E 1        NIBHEX 1                   Non-programmable!

0004F D20      CON(3) 45              02  Do FORTH with parameter
00052 0000     REL(5) =FORTHX             from BASIC
      0
00057 D        NIBHEX D

00058 E10      CON(3)  30             03  BASIC-->FORTH with
0005B 0000     REL(5)  =FORTHI             integer result
      0
00060 F        NIBHEX  F

00061 F00      CON(3)  15             04  BASIC-->FORTH with
00064 0000     REL(5)  =FORTHF             floating point result
      0
00069 F        NIBHEX  F


0006A 000      CON(3)  0              05  BASIC-->FORTH with
0006D 0000     REL(5)  =FORTH$             string result
      0
00072 F        NIBHEX  F
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER


```
                    STITLE T e x t   T a b l e

               *  Text Table

00073          TxTbSt                      Text table start
               *
00073 B        NIBHEX  B
00074 64F4     NIBASC  \FORTH$\
      2545
      8442
00080 50       NIBHEX  50

00082 B        NIBHEX  B
00083 64F4     NIBASC  \FORTHF\
      2545
      8464
0008F 40       NIBHEX  40

00091 B        NIBHEX  B
00092 64F4     NIBASC  \FORTHI\
      2545
      8494
0009E 30       NIBHEX  30

000A0 B        NIBHEX B
000A1 64F4     NIBASC \FORTHX\
      2545
      8485
000AD 20       NIBHEX 20

000AF 9        NIBHEX 9                 Invoke FORTH system
000B0 64F4     NIBASC \FORTH\
      2545
      84
000BA 10       NIBHEX 10

000BC 1FF   TxTbEn NIBHEX 1FF           Text termination
```

```
                STITLE  ACTIVE TABLE
*************************************************************
*
* The location ACTIVE in the file FORTHRAM is used by
* the system to keep track of where it has been in
* the many and convoluted paths through BASIC and FORTH.
* Here are the meaningful values of ACTIVE:
*
*      -2       Flag set while handling the error poll,
*               before we send another error poll, so we
*               can ignore this second-order poll.
*      -1       We went to sleep in FORTH, and have now
*               awakened, realized this in BASPOL, and set
*               the new value in forpon, which will be
*               checked and reset in RESTRT.
*       0       FORTH has not been called, or it
*               was left with a BYE.
*       1       Normal condition; FORTH environment
*               is active.
*       2       BASICI,BASICF or  BASIC$ is running; they
*               can't return via BASPOL, so if
*               MainLoop poll occurred we have to go to
*               ABORT.
*       3       BASICX is running the BASIC environment,
*               ready to return via the MainLoop poll(BASPOL).
*     [4,5]     There's a check in ERR02 which cares about 2&3
*               but would also catch 4&5.
*      99       We just went to sleep by calling power-off
*               in EXPECT, and have just awakened.
*
*************************************************************
```



```
                STITLE  ERROR MESSAGE TABLE
000BF       MsgTbl
000BF 10            CON(2)   1       Min message #
000C1 04            CON(2)   64      Max message #
                *
                * NOTE: First message has to be of length which is
                * multiple of 16, i.e. next nibble right here has
                * to be zero, because of bug in message-driver code.
                *
                *eTMAD  EQU     36
000C3 04            CON(2)  64
000C5 42            CON(2)  36          Message number  36
000C7 A             CON(1)  10
000C8 47F6          NIBASC \too many\
      F602
      D616
      E697
000D8 0214          NIBASC \ AS\            changed to upper case
      35
000DE A             CON(1)  10              2/6/84
000DF 3494          NIBASC \CII char\
      9402
      3686
      1627
000EF 3702          NIBASC \s p\
      07
000F5 5             CON(1)   5
000F6 2756          NIBASC \resent\
      3756
      E647
00102 C             CON(1)  12
                *
                *eRROR  EQU      0          GOES EVERYWHERE
00103 E0            CON(2)  14
00105 00            CON(2)   0          Message number   0
00107 3             CON(1)   3
00108 6445          NIBASC \FTH \
      8402
00110 C             CON(1)  12
                *
                *eNEND  EQU      1          no ending something
00111 A1            CON(2)  26
00113 10            CON(2)   1          Message number   1
00115 9             CON(1)   9
00116 E6F6          NIBASC \no endin\
      0256
      E646
      96E6
00126 7602          NIBASC \g \
0012A C             CON(1)  12
                *
                *eNOTF  EQU      2          xxx Not Found
0012B A0            CON(2)  10
0012D 20            CON(2)   2          Message number   2
0012F F3            CON(2)  63
```

```
160 00131 E           CON(1)  14
161 00132 00          CON(2) =eNFOUN
162 00134 C           CON(1)  12
163           *
164           *eNO;   EQU      3          no ending ;
165 00135 B0          CON(2)  11
166 00137 30          CON(2)   3          Message number   3
167 00139 D           CON(1)  13
168 0013A 10          CON(2) =eNEND
169 0013C 0           CON(1)   0
170 0013D B3          NIBASC \;\
171 0013F C           CON(1)  12
172           *
173           *eNOEP  EQU      4          no ending )
174 00140 B0          CON(2)  11
175 00142 40          CON(2)   4          Message number   4
176 00144 D           CON(1)  13
177 00145 10          CON(2) =eNEND
178 00147 0           CON(1)   0
179 00148 92          NIBASC \)\
180 0014A C           CON(1)  12
181           *
182           *eNO"   EQU      5          no ending "
183 0014B B0          CON(2)  11
184 0014D 50          CON(2)   5          Message number   5
185 0014F D           CON(1)  13
186 00150 10          CON(2) =eNEND
187 00152 0           CON(1)   0
188 00153 22          NIBASC \"\
189 00155 C           CON(1)  12
190           *
191           *eLT1   EQU      6
192 00156 12          CON(2)  33
193 00158 60          CON(2)   6          Message number   6
194 0015A F3          CON(2)  63
195 0015C B           CON(1)  11
196 0015D B           CON(1)  11
197 0015E 1627        NIBASC \argument\
          7657
          D656
          E647
198 0016E 02C3        NIBASC \ < 1\
          0213
199 00176 C           CON(1)  12
200           *
201           *eNDEF  EQU      7
202 00177 63          CON(2)  54
203 00179 70          CON(2)   7          Message number   7
204 0017B A           CON(1)  10
205 0017C 4656        NIBASC \definiti\
          6696
          E696
          4796
206 0018C F6E6        NIBASC \on \
          02
```



```
207 00192 B           CON(1)  11
208 00193 B           CON(1)  11
209 00194 E6F6        NIBASC \not fini\
          4702
          6696
          E696
210 001A4 3786        NIBASC \shed\
          5646
211 001AC C           CON(1)  12
212           *
213           *eDFUL  EQU      8
214 001AD 52          CON(2)  37
215 001AF 80          CON(2)   8          Message number   8
216 001B1 B           CON(1)  11
217 001B2 E           CON(1)  14
218 001B3 4696        NIBASC \dictiona\
          3647
          96F6
          E616
219 001C3 2797        NIBASC \ry full\
          0266
          57C6
          C6
220 001D1 C           CON(1)  12
221           *
222           *eCOMP  EQU      9
223 001D2 12          CON(2)  33
224 001D4 90          CON(2)   9          Message number   9
225 001D6 F3          CON(2)  63
226 001D8 B           CON(1)  11
227 001D9 B           CON(1)  11
228 001DA 36F6        NIBASC \compile \
          D607
          96C6
          5602
229 001EA F6E6        NIBASC \only\
          C697
230 001F2 C           CON(1)  12
231           *
232           *ePILE  EQU      10
233 001F3 C1          CON(2)  28
234 001F5 A0          CON(2)  10          Message number  10
235 001F7 F3          CON(2)  63
236 001F9 9           CON(1)   9
237 001FA 8405        NIBASC \HPIL err\
          94C4
          0256
          2727
238 0020A F627        NIBASC \or\
239 0020E C           CON(1)  12
240           *
241           *eRDNL  EQU      11
242 0020F C3          CON(2)  60
243 00211 B0          CON(2)  11          Message number  11
244 00213 A           CON(1)  10
```

```
245 00214 1647          NIBASC \attempte\
          4756
          D607
          4756
246 00224 4602          NIBASC \d t\
          47
247 0022A B             CON(1)  11
248 0022B E             CON(1)  14
249 0022C F602          NIBASC \o redefi\
          2756
          4656
          6696
250 0023C E656          NIBASC \ne null\
          02E6
          57C6
          C6
251 0024A C             CON(1)  12
252             *
253             *eBADD  EQU     12
254 0024B F2            CON(2)  47
255 0024D C0            CON(2)  12              Message number  12
256 0024F A             CON(1)  10
257 00250 2616          NIBASC \bad dict\
          4602
          4696
          3647
258 00260 96F6          NIBASC \ion\
          E6
259 00266 8             CON(1)   8
260 00267 1627          NIBASC \ary entr\
          9702
          56E6
          4727
261 00277 97            NIBASC \y\
262 00279 C             CON(1)  12
263             *
264             *ePDIC  EQU     13
265 0027A 63            CON(2)  54
266 0027C D0            CON(2)  13              Message number  13
267 0027E A             CON(1)  10
268 0027F 96E6          NIBASC \in prote\
          0207
          27F6
          4756
269 0028F 3647          NIBASC \cte\
          56
270 00295 B             CON(1)  11
271 00296 B             CON(1)  11
272 00297 4602          NIBASC \d dictio\
          4696
          3647
          96F6
273 002A7 E616          NIBASC \nary\
          2797
274 002AF C             CON(1)  12
```



```
275             *
276             *eEMPT  EQU     14
277 002B0 C1            CON(2)  28
278 002B2 E0            CON(2)  14              Message number  14
279 002B4 A             CON(1)  10
280 002B5 56D6          NIBASC \empty st\
          0747
          9702
          3747
281 002C5 1636          NIBASC \ack\
          B6
282 002CB C             CON(1)  12
283             *
284             *eFULL  EQU     15
285 002CC A1            CON(2)  26
286 002CE F0            CON(2)  15              Message number  15
287 002D0 9             CON(1)   9
288 002D1 6657          NIBASC \full sta\
          C6C6
          0237
          4716
289 002E1 36B6          NIBASC \ck\
290 002E5 C             CON(1)  12
291             *
292             *eNREC  EQU     16
293 002E6 52            CON(2)  37
294 002E8 01            CON(2)  16              Message number  16
295 002EA F3            CON(2)  63
296 002EC B             CON(1)  11
297 002ED D             CON(1)  13
298 002EE E6F6          NIBASC \not reco\
          4702
          2756
          36F6
299 002FE 76E6          NIBASC \gnized\
          96A7
          5646
300 0030A C             CON(1)  12
301             *
302             *eCOND  EQU     17
303 0030B 83            CON(2)  56
304 0030D 11            CON(2)  17              Message number  17
305 0030F F3            CON(2)  63
306 00311 A             CON(1)  10
307 00312 36F6          NIBASC \conditio\
          E646
          9647
          96F6
308 00322 E616          NIBASC \nal\
          C6
309 00328 B             CON(1)  11
310 00329 B             CON(1)  11
311 0032A 3702          NIBASC \s not pa\
          E6F6
          4702
```

```
          0716
  312 0033A 9627        NIBASC \ired\
          5646
  313 00342 C           CON(1)  12
  314           *
  315           *eNOFTH EQU     18
  316 00343 C3          CON(2)  60
  317 00345 21          CON(2)  18          Message number  18
  318 00347 A           CON(1)  10
  319 00348 64F4        NIBASC \FORTHRAM\
          2545
          8425
          14D4
  320 00358 0266        NIBASC \ fi\
          96
  321 0035E B           CON(1)  11
  322 0035F E           CON(1)  14
  323 00360 C656        NIBASC \le not i\
          02E6
          F647
          0296
  324 00370 E602        NIBASC \n place\
          07C6
          1636
          56
  325 0037E C           CON(1)  12
  326           *
  327           *eBFIL  EQU     19
  328 0037F A0          CON(2)  10
  329 00381 31          CON(2)  19          Message number  19
  330 00383 F3          CON(2)  63
  331 00385 E           CON(1)  14
  332 00386 00          CON(2) =eFSPEC
  333 00388 C           CON(1)  12
  334           *
  335           *eUNKO  EQU     20
  336 00389 32          CON(2)  35
  337 0038B 41          CON(2)  20          Message number  20
  338 0038D B           CON(1)  11
  339 0038E D           CON(1)  13
  340 0038F 57E6        NIBASC \unknown \
          B6E6
          F677
          E602
  341 0039F F607        NIBASC \opcode\
          36F6
          4656
  342 003AB C           CON(1)  12
  343           *
  344           *eGRYR  EQU     21
  345 003AC 83          CON(2)  56
  346 003AE 51          CON(2)  21          Message number  21
  347 003B0 A           CON(1)  10
  348 003B1 74F4        NIBASC \GOYES or\
          9554
```



```
          3502
          F627
  349 003C1 0225        NIBASC \ RT\
          45
  350 003C7 B           CON(1)  11
  351 003C8 C           CON(1)  12
  352 003C9 E495        NIBASC \NYES req\
          5435
          0227
          5617
  353 003D9 5796        NIBASC \uired\
          2756
          46
  354 003E3 C           CON(1)  12
  355           *
  356           *eMISL  EQU     22
  357 003E4 13          CON(2)  49
  358 003E6 61          CON(2)  22          Message number  22
  359 003E8 A           CON(1)  10
  360 003E9 D696        NIBASC \missing/\
          3737
          96E6
          76F2
  361 003F9 96C6        NIBASC \ill\
          C6
  362 003FF 9           CON(1)   9
  363 00400 5676        NIBASC \egal lab\
          16C6
          02C6
          1626
  364 00410 56C6        NIBASC \el\
  365 00414 C           CON(1)  12
  366           *
  367           *eNLST  EQU     23
  368 00415 83          CON(2)  56
  369 00417 71          CON(2)  23          Message number  23
  370 00419 A           CON(1)  10
  371 0041A 96E6        NIBASC \invalid \
          6716
          C696
          4602
  372 0042A C696        NIBASC \lis\
          37
  373 00430 B           CON(1)  11
  374 00431 C           CON(1)  12
  375 00432 4796        NIBASC \ting arg\
          E676
          0216
          2776
  376 00442 57D6        NIBASC \ument\
          56E6
          47
  377 0044C C           CON(1)  12
  378           *
  379           *eCCMS  EQU     24
```

```
380 0044D 13          CON(2)  49
381 0044F 81          CON(2)  24          Message number  24
382 00451 A           CON(1)  10
383 00452 96E6        NIBASC \invalid \
          6716
          C696
          4602
384 00462 1757        NIBASC \quo\
          F6
385 00468 9           CON(1)   9
386 00469 4756        NIBASC \ted stri\
          4602
          3747
          2796
387 00479 E676        NIBASC \ng\
388 0047D C           CON(1)  12
389           *
390           *eWNUQ  EQU     25
391 0047E 83          CON(2)  56
392 00480 91          CON(2)  25          Message number  25
393 00482 A           CON(1)  10
394 00483 7716        NIBASC \warning:\
          27E6
          96E6
          76A3
395 00493 0277        NIBASC \ wo\
          F6
396 00499 B           CON(1)  11
397 0049A C           CON(1)  12
398 0049B 2746        NIBASC \rd not u\
          02E6
          F647
          0257
399 004AB E696        NIBASC \nique\
          1757
          56
400 004B5 C           CON(1)  12
401           *
402           *eIFNS  EQU     26
403 004B6 C3          CON(2)  60
404 004B8 A1          CON(2)  26          Message number  26
405 004BA A           CON(1)  10
406 004BB 96E6        NIBASC \invalid \
          6716
          C696
          4602
407 004CB 6696        NIBASC \fil\
          C6
408 004D1 B           CON(1)  11
409 004D2 E           CON(1)  14
410 004D3 56E6        NIBASC \ename sp\
          16D6
          5602
          3707
411 004E3 5636        NIBASC \ecifier\
```



```
          9666
          9656
          27
412 004F1 C           CON(1)  12
413           *
414           *eILFS  EQU     27
415 004F2 D2          CON(2)  45
416 004F4 B1          CON(2)  27          Message number  27
417 004F6 A           CON(1)  10
418 004F7 96C6        NIBASC \illegal \
          C656
          7616
          C602
419 00507 77F6        NIBASC \wor\
          27
420 0050D 7           CON(1)   7
421 0050E 4602        NIBASC \d select\
          3756
          C656
          3647
422 0051E C           CON(1)  12
423           *
424           *eILEN  EQU     28
425 0051F 63          CON(2)  54
426 00521 C1          CON(2)  28          Message number  28
427 00523 A           CON(1)  10
428 00524 A657        NIBASC \jump or \
          D607
          02F6
          2702
429 00534 6716        NIBASC \val\
          C6
430 0053A B           CON(1)  11
431 0053B B           CON(1)  11
432 0053C 5756        NIBASC \ue too l\
          0247
          F6F6
          02C6
433 0054C 1627        NIBASC \arge\
          7656
434 00554 C           CON(1)  12
435           *
436           *eRPTS  EQU     29
437 00555 64          CON(2)  70
438 00557 D1          CON(2)  29          Message number  29
439 00559 A           CON(1)  10
440 0055A E656        NIBASC \needs pr\
          5646
          3702
          0727
441 0056A 5667        NIBASC \evi\
          96
442 00570 A           CON(1)  10
443 00571 F657        NIBASC \ous test\
          3702
```

```
                4756
                3747
    444 00581   0296        NIBASC \ in\
                E6
    445 00587   8           CON(1)   8
    446 00588   3747        NIBASC \structio\
                2757
                3647
                96F6
    447 00598   E6          NIBASC \n\
    448 0059A   C           CON(1)  12
    449             *
    450             *eILPT  EQU     30
    451 0059B   83          CON(2)  56
    452 0059D   E1          CON(2)  30          Message number  30
    453 0059F   A           CON(1)  10
    454 005A0   96C6        NIBASC \illegal \
                C656
                7616
                C602
    455 005B0   07F6        NIBASC \poi\
                96
    456 005B6   B           CON(1)  11
    457 005B7   C           CON(1)  12
    458 005B8   E647        NIBASC \nter pos\
                5627
                0207
                F637
    459 005C8   9647        NIBASC \ition\
                96F6
                E6
    460 005D2   C           CON(1)  12
    461             *
    462             *eILSB  EQU     31
    463 005D3   B2          CON(2)  43
    464 005D5   F1          CON(2)  31          Message number  31
    465 005D7   A           CON(1)  10
    466 005D8   96C6        NIBASC \illegal \
                C656
                7616
                C602
    467 005E8   3747        NIBASC \sta\
                16
    468 005EE   6           CON(1)   6
    469 005EF   4757        NIBASC \tus bit\
                3702
                2696
                47
    470 005FD   C           CON(1)  12
    471             *
    472             *eIDPA  EQU     32
    473 005FE   E3          CON(2)  62
    474 00600   02          CON(2)  32          Message number  32
    475 00602   A           CON(1)  10
    476 00603   96C6        NIBASC \illegal \
```



```
                C656
                7616
                C602
    477 00613   4607        NIBASC \dp \
                02
    478 00619   B           CON(1)  11
    479 0061A   F           CON(1)  15
    480 0061B   1627        NIBASC \arithmet\
                9647
                86D6
                5647
    481 0062B   9636        NIBASC \ic value\
                0267
                16C6
                5756
    482 0063B   C           CON(1)  12
    483             *
    484             *eILTV  EQU     33
    485 0063C   33          CON(2)  51
    486 0063E   12          CON(2)  33          Message number  33
    487 00640   A           CON(1)  10
    488 00641   96C6        NIBASC \illegal \
                C656
                7616
                C602
    489 00651   4727        NIBASC \tra\
                16
    490 00657   A           CON(1)  10
    491 00658   E637        NIBASC \nsfer va\
                6656
                2702
                6716
    492 00668   C657        NIBASC \lue\
                56
    493 0066E   C           CON(1)  12
    494             *
    495             *eNHDP  EQU     34
    496 0066F   24          CON(2)  66
    497 00671   22          CON(2)  34          Message number  34
    498 00673   A           CON(1)  10
    499 00674   E6F6        NIBASC \non-hexa\
                E6D2
                8656
                8716
    500 00684   4656        NIBASC \dec\
                36
    501 0068A   A           CON(1)  10
    502 0068B   96D6        NIBASC \imal dig\
                16C6
                0246
                9676
    503 0069B   9647        NIBASC \it \
                02
    504 006A1   6           CON(1)   6
    505 006A2   0727        NIBASC \present\
```

```
          5637
          56E6
          47
506 006B0 C           CON(1)  12
507           *
508           *eTMHD  EQU     35
509 006B1 E3          CON(2)  62
510 006B3 32          CON(2)  35          Message number  35
511 006B5 A           CON(1)  10
512 006B6 47F6        NIBASC \too many\
          F602
          D616
          E697
513 006C6 0286        NIBASC \ he\
          56
514 006CC B           CON(1)  11
515 006CD F           CON(1)  15
516 006CE 8702        NIBASC \x digits\
          4696
          7696
          4737
517 006DE 0207        NIBASC \ present\
          2756
          3756
          E647
518 006EE C           CON(1)  12
519           *
520           *eUNRL  EQU     37
521 006EF B2          CON(2)  43
522 006F1 52          CON(2)  37          Message number  37
523 006F3 A           CON(1)  10
524 006F4 57E6        NIBASC \unrecogn\
          2756
          36F6
          76E6
525 00704 96A7        NIBASC \ize\
          56
526 0070A 6           CON(1)   6
527 0070B 4602        NIBASC \d label\
          C616
          2656
          C6
528 00719 C           CON(1)  12
529           *
530           *eMMPR  EQU     38
531 0071A 33          CON(2)  51
532 0071C 62          CON(2)  38          Message number  38
533 0071E A           CON(1)  10
534 0071F D096        NIBASC \mismatch\
          37D6
          1647
          3686
535 0072F 5646        NIBASC \ed \
          02
536 00735 A           CON(1)  10
```



```
537 00736 0716        NIBASC \parenthe\
          2756
          E647
          8656
538 00746 3756        NIBASC \ses\
          37
539 0074C C           CON(1)  12
540           *
541           *eUCIE  EQU     39
542 0074D B2          CON(2)  43
543 0074F 72          CON(2)  39          Message number  39
544 00751 A           CON(1)  10
545 00752 96C6        NIBASC \illegal \
          C656
          7616
          C602
546 00762 5687        NIBASC \exp\
          07
547 00768 6           CON(1)   6
548 00769 2756        NIBASC \ression\
          3737
          96F6
          E6
549 00777 C           CON(1)  12
550           *
551           *eECIE  EQU     40
552 00778 64          CON(2)  70
553 0077A 82          CON(2)  40          Message number  40
554 0077C A           CON(1)  10
555 0077D 5687        NIBASC \excess c\
          3656
          3737
          0236
556 0078D 8616        NIBASC \har\
          27
557 00793 A           CON(1)  10
558 00794 1636        NIBASC \acters i\
          4756
          2737
          0296
559 007A4 E602        NIBASC \n e\
          56
560 007AA 8           CON(1)   8
561 007AB 8707        NIBASC \xpressio\
          2756
          3737
          96F6
562 007BB E6          NIBASC \n\
563 007BD C           CON(1)  12
564           *
565           *eDLAB  EQU     41
566 007BE 52          CON(2)  37
567 007C0 92          CON(2)  41          Message number  41
568 007C2 B           CON(1)  11
569 007C3 E           CON(1)  14
```

```
570 007C4 4657          NIBASC \duplicat\
          07C6
          9636
          1647
571 007D4 5602          NIBASC \e label\
          C616
          2656
          C6
572 007E2 C             CON(1)  12
573             *
574             *eSYTF  EQU     42
575 007E3 92            CON(2)  41
576 007E5 A2            CON(2)  42          Message number  42
577 007E7 A             CON(1)  10
578 007E8 3797          NIBASC \symbol t\
          D626
          F6C6
          0247
579 007F8 1626          NIBASC \abl\
          C6
580 007FE 5             CON(1)   5
581 007FF 5602          NIBASC \e full\
          6657
          C6C6
582 0080B C             CON(1)  12
583             *
584             *eRANGE EQU     43
585 0080C A3            CON(2)  58
586 0080E B2            CON(2)  43          Message number  43
587 00810 A             CON(1)  10
588 00811 1646          NIBASC \address \
          4627
          5637
          3702
589 00821 E6F6          NIBASC \not\
          47
590 00827 B             CON(1)  11
591 00828 D             CON(1)  13
592 00829 0296          NIBASC \ inside \
          E637
          9646
          5602
593 00839 1602          NIBASC \a file\
          6696
          C656
594 00845 C             CON(1)  12
595             *
596             *eLFUL  EQU     44
597 00846 92            CON(2)  41
598 00848 C2            CON(2)  44          Message number  44
599 0084A A             CON(1)  10
600 0084B C696          NIBASC \listing \
          3747
          96E6
          7602
```



```
601 0085B 6696          NIBASC \fil\
          C6
602 00861 5             CON(1)   5
603 00862 5602          NIBASC \e full\
          6657
          C6C6
604 0086E C             CON(1)  12
605             *
606             *eDUPFL EQU     45
607 0086F A0            CON(2)  10
608 00871 D2            CON(2)  45          Message number  45
609 00873 F3            CON(2)  63
610 00875 E             CON(1)  14
611 00876 00            CON(2) =eFEXSI
612 00878 C             CON(1)  12
613             *
614             *eBADPM EQU     46
615 00879 52            CON(2)  37
616 0087B E2            CON(2)  46          Message number  46
617 0087D F3            CON(2)  63
618 0087F B             CON(1)  11
619 00880 D             CON(1)  13
620 00881 2616          NIBASC \bad para\
          4602
          0716
          2716
621 00891 D656          NIBASC \meters\
          4756
          2737
622 0089D C             CON(1)  12
623             *
624             *eNRFA  EQU     47
625 0089E 64            CON(2)  70
626 008A0 F2            CON(2)  47          Message number  47
627 008A2 A             CON(1)  10
628 008A3 E6F6          NIBASC \not enou\
          4702
          56E6
          F657
629 008B3 7686          NIBASC \gh \
          02
630 008B9 A             CON(1)  10
631 008BA D656          NIBASC \memory f\
          D6F6
          2797
          0266
632 008CA F627          NIBASC \or \
          02
633 008D0 8             CON(1)   8
634 008D1 1637          NIBASC \assemble\
          3756
          D626
          C656
635 008E1 27            NIBASC \r\
636 008E3 C             CON(1)  12
```

```
637                *
638                *eLFNT  EQU     48
639 008E4 13              CON(2)  49
640 008E6 03              CON(2)  48          Message number  48
641 008E8 A               CON(1)  10
642 008E9 C696            NIBASC \listing \
          3747
          96E6
          7602
643 008F9 6696            NIBASC \fil\
          C6
644 008FF 9               CON(1)   9
645 00900 5602            NIBASC \e not TE\
          E6F6
          4702
          4554
646 00910 8545            NIBASC \XT\
647 00914 C               CON(1)  12
648                *
649                *eIVLF  EQU     49
650 00915 F2              CON(2)  47
651 00917 13              CON(2)  49          Message number  49
652 00919 A               CON(1)  10
653 0091A 96E6            NIBASC \invalid \
          6716
          C696
          4602
654 0092A C696            NIBASC \lis\
          37
655 00930 8               CON(1)   8
656 00931 4796            NIBASC \ting fil\
          E676
          0266
          96C6
657 00941 56              NIBASC \e\
658 00943 C               CON(1)  12
659                *
660                *eCOSF  EQU     50
661 00944 63              CON(2)  54
662 00946 23              CON(2)  50          Message number  50
663 00948 A               CON(1)  10
664 00949 3616            NIBASC \cannot o\
          E6E6
          F647
          02F6
665 00959 0756            NIBASC \pen\
          E6
666 0095F B               CON(1)  11
667 00960 B               CON(1)  11
668 00961 0237            NIBASC \ source \
          F657
          2736
          5602
669 00971 6696            NIBASC \file\
          C656
```



```
670 00979 C               CON(1)  12
671                *
672                *eMMFT  EQU     51
673 0097A C3              CON(2)  60
674 0097C 33              CON(2)  51          Message number  51
675 0097E A               CON(1)  10
676 0097F D696            NIBASC \missing/\
          3737
          96E6
          76F2
677 0098F D657            NIBASC \mul\
          C6
678 00995 B               CON(1)  11
679 00996 E               CON(1)  14
680 00997 4796            NIBASC \tiple fi\
          07C6
          5602
          6696
681 009A7 C656            NIBASC \le type\
          0247
          9707
          56
682 009B5 C               CON(1)  12
683                *
684                *eCONF  EQU     52
685 009B6 12              CON(2)  33
686 009B8 43              CON(2)  52          Message number  52
687 009BA B               CON(1)  11
688 009BB C               CON(1)  12
689 009BC 34F6            NIBASC \Configur\
          E666
          9676
          5727
690 009CC 1647            NIBASC \ation\
          96F6
          E6
691 009D6 C               CON(1)  12
692                *
693                *eSWF   EQU     53
694 009D7 92              CON(2)  41
695 009D9 53              CON(2)  53          Message number  53
696 009DB F3              CON(2)  63
697 009DD B               CON(1)  11
698 009DE F               CON(1)  15
699 009DF 3747            NIBASC \string w\
          2796
          E676
          0277
700 009EF F6E6            NIBASC \on't fit\
          7247
          0266
          9647
701 009FF C               CON(1)  12
702                *
703                *eNCUR  EQU     54
```

```
704 00A00 C3          CON(2)  60
705 00A02 63          CON(2)  54          Message number  54
706 00A04 F3          CON(2)  63
707 00A06 A           CON(1)  10
708 00A07 E6F6        NIBASC \not in c\
          4702
          96E6
          0236
709 00A17 5727        NIBASC \urr\
          27
710 00A1D B           CON(1)  11
711 00A1E D           CON(1)  13
712 00A1F 56E6        NIBASC \ent voca\
          4702
          67F6
          3616
713 00A2F 2657        NIBASC \bulary\
          C616
          2797
714 00A3B C           CON(1)  12
715             *
716             *eCNLD  EQU     55
717 00A3C E1          CON(2)  30
718 00A3E 73          CON(2)  55          Message number  55
719 00A40 F3          CON(2)  63
720 00A42 A           CON(1)  10
721 00A43 3616        NIBASC \cannot l\
          E6E6
          F647
          02C6
722 00A53 F616        NIBASC \oad\
          46
723 00A59 C           CON(1)  12
724             *
725             *eASAB  EQU     56
726 00A5A 92          CON(2)  41
727 00A5C 83          CON(2)  56          Message number  56
728 00A5E A           CON(1)  10
729 00A5F 1637        NIBASC \assemble\
          3756
          D626
          C656
730 00A6F 2702        NIBASC \r a\
          16
731 00A75 5           CON(1)   5
732 00A76 26F6        NIBASC \borted\
          2747
          5646
733 00A82 C           CON(1)  12
734             *
735             *eCREQ  EQU     57
736 00A83 13          CON(2)  49
737 00A85 93          CON(2)  57          Message number  57
738 00A87 A           CON(1)  10
739 00A88 3616        NIBASC \cannot r\
```



```
          E6E6
          F647
          0227
740 00A98 5637        NIBASC \eso\
          F6
741 00A9E 9           CON(1)   9
742 00A9F C667        NIBASC \lve equa\
          5602
          5617
          5716
743 00AAF 4756        NIBASC \te\
744 00AB3 C           CON(1)  12
745             *
746             *ePSTS  EQU     58
747 00AB4 B2          CON(2)  43
748 00AB6 A3          CON(2)  58          Message number  58
749 00AB8 A           CON(1)  10
750 00AB9 0716        NIBASC \pagesize\
          7656
          3796
          A756
751 00AC9 0247        NIBASC \ to\
          F6
752 00ACF 6           CON(1)   6
753 00AD0 F602        NIBASC \o small\
          37D6
          16C6
          C6
754 00ADE C           CON(1)  12
755             *
756             *eNODO  EQU     59
757 00ADF B2          CON(2)  43
758 00AE1 B3          CON(2)  59          Message number  59
759 00AE3 A           CON(1)  10
760 00AE4 E6F6        NIBASC \no DO be\
          0244
          F402
          2656
761 00AF4 66F6        NIBASC \for\
          27
762 00AFA 6           CON(1)   6
763 00AFB 5602        NIBASC \e LEAVE\
          C454
          1465
          54
764 00B09 C           CON(1)  12
765             *
766             *eCASE  EQU     60
767 00B0A 33          CON(2)  51
768 00B0C C3          CON(2)  60          Message number  60
769 00B0E A           CON(1)  10
770 00B0F 96C6        NIBASC \illegal \
          C656
          7616
          C602
```

```
771 00B1F 3414          NIBASC \CAS\
          35
772 00B25 A             CON(1)  10
773 00B26 5402          NIBASC \E struct\
          3747
          2757
          3647
774 00B36 5727          NIBASC \ure\
          56
775 00B3C C             CON(1)  12
776             *
777             *eRLAB  EQU     61
778 00B3D 44            CON(2)  68
779 00B3F D3            CON(2)  61              Message number  61
780 00B41 A             CON(1)  10
781 00B42 2756          NIBASC \restrict\
          3747
          2796
          3647
782 00B52 5646          NIBASC \ed \
          02
783 00B58 A             CON(1)  10
784 00B59 C616          NIBASC \label Fi\
          2656
          C602
          6496
785 00B69 C456          NIBASC \LeN\
          E4
786 00B6F 7             CON(1)   7
787 00B70 4602          NIBASC \d exists\
          5687
          9637
          4737
788 00B80 C             CON(1)  12
789             *
790             *eREENT EQU     62
791 00B81 52            CON(2)  37
792 00B83 E3            CON(2)  62              Message number  62
793 00B85 B             CON(1)  11
794 00B86 E             CON(1)  14
795 00B87 02E6          NIBASC \ not re-\
          F647
          0227
          56D2
796 00B97 56E6          NIBASC \entrant\
          4727
          16E6
          47
797 00BA5 C             CON(1)  12
798             *
799             *eBASCX EQU     63
800 00BA6 31            CON(2)  19
801 00BA8 F3            CON(2)  63              Message number  63
802 00BAA 4             CON(1)   4
803 00BAB 2414          NIBASC \BASIC\
```



```
          3594
          34
804 00BB5 D             CON(1)  13
805 00BB6 E3            CON(2) =eREENT
806 00BB8 C             CON(1)  12
807             *
808             *eFORTX EQU     64
809 00BB9 31            CON(2)  19
810 00BBB 04            CON(2)  64              Message number  64
811 00BBD 4             CON(1)   4
812 00BBE 64F4          NIBASC \FORTH\
          2545
          84
813 00BC8 D             CON(1)  13
814 00BC9 E3            CON(2) =eREENT
815 00BCB C             CON(1)  12
816             *
817 00BCC FF            NIBHEX FF               Table terminator
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER

```
818                    STITLE POLL_HANDLING
819                ********************************************************
820                * The following code is the FORTH ROM poll handler.     *
821                * On entry B(B) = process#.  The FORTH ROM will respond *
822                * to the following process numbers:                     *
823                *                                          CODE#(HEX)*
824                * pERROR (slow poll) - poll for errors        F2        *
825                * pVER$ - VER$ function poll                            *
826                * pMNLP - main loop poll                     FA        *
827                * pMEM - insufficient memory error poll      F1        *
828                * pFTYPE - unknown file type poll            2D        *
829                * pCONFG - configuration poll                FB        *
830                *                                                       *
831                ********************************************************
832
833
834 00BCE 04    POLHND SETHEX
835 00BD0 20           P=     0
836 00BD2 969          ?B=0   B              VER$ poll?
837 00BD5 23           GOYES  hVER$0         Yes.
838 00BD7 3100         LC(2)  =pMNLP
839 00BDB 961          ?B=C   B              MAINLOOP poll?
840 00BDE 95           GOYES  baspol         Yes.
841 00BE0 3100         LC(2)  =pFTYPE
842 00BE4 961          ?B=C   B              CAT poll?
843 00BE7 45           GOYES  catpol         Yes.
844 00BE9 3100         LC(2)  =pERROR
845 00BED 961          ?B=C   B              ERROR?
846 00BF0 F4           GOYES  errhnd         Yes.
847 00BF2 3100         LC(2)  =pMEM
848 00BF6 961          ?B=C   B              INSUFFICIENT MEMORY poll?
849 00BF9 A4           GOYES  errMEM         Yes.
850 00BFB 3100         LC(2)  =pCONFG
851 00BFF 961          ?B=C   B              CONFIGURATION poll?
852 00C02 16           GOYES  config         Yes.
853
854 00C04 503          GONC   hVER$2         No.  To hVER$2 w/carry clear.
855 00C07 11B  hVER$0  C=R3
856 00C0A 135          D1=C
857 00C0D 112          A=R2
858 00C10 1CD          D1=D1- (VER$en)-(VER$st)-2
859 00C13 137          CD1EX
860 00C16 8B6          ?A>C   A
861 00C19 C1           GOYES  hVER$1
862 00C1B 135          D1=C
863 00C1E 10B          R3=C
864 00C21 3D14 VER$st  LCASC  \ FTH:1A\
          13A3
          8445
          6402
865 00C31 15DD VER$en  DAT1=C (VER$en)-(VER$st)-2
866 00C35      hVER$1
867 00C35      hVER$2
868 00C35 00           RTNSXM
869 00C37 62A0 baspol  GOTO    BASPOL
```



```
870 00C3B 62E1 catpol  GOTO    CATPOL
871 00C3F 6B42 errhnd  GOTO    ERRHND
872
873            * MEMERR poll is problematic because it returns
874            * to mainloop instead of caller of MFERR
875            * So we're setting this flag in oERR? to tell
876            * MAINLOOP(baspol)poll to abort.
877 00C43 7BF1 errMEM  GOSUB   FTHRM?
878 00C47 5DE          GONC    hVER$2        IF NO FORTHRAM, RTNSXM
879 00C4A 7322         GOSUB   GOOD?         FORTHRAM in position?
880 00C4E 56E          GONC    hVER$2        no, RTNSXM
881 00C51 1BBB         D0=(5)  =oERR?
          BF2
882 00C58 D2           C=0     A
883 00C5A E6           C=C+1   A
884 00C5C 144          DAT0=C  A             ERR?=1
885 00C5F 6C42         GOTO    ERRHX0
886
```

```
887                     EJECT
888                 ****************************************
889                 *
890                 * CONFIGURATION poll handler
891                 *
892                 * A user may have pulled or added modules
893                 * while the machine was turned off!
894                 * This code will allow us to ensure that
895                 * our RAM file is put back where it should
896                 * be!
897                 *
898                 * So, given that FORTHRAM is the first file
899                 * in memory and that it is of the right type,
900                 * this rtne allows us to assume that the
901                 * addressees in FORTHRAM (i.e., ACTIVE) will
902                 * always be correct.
903                 *
904                 ****************************************
905
906 00C63 7BD1 config  GOSUB   FTHRM?          if this rtne returns w/
907                 *                          with carry clear FORTHRAM
908                 *                          is not right type or in
909                 *                          right place, so just rtnsxm
910 00C67 5DC         GONC    hVER$2
911
912                 *
913                 * now we know that FORTHRAM is the first file in memory
914                 * and that it is of the correct type, let's check to see
915                 * if it has moved since that last time we were in FORTH!
916                 *
917 00C6A 16F         D0=D0+  16              pt to old addr of FORTHRAM
918 00C6D 164         D0=D0+  5               which we keep right after
919                 *                          the file header
920 00C70 142         A=DAT0  A               read old addr
921 00C73 1B00       D0=(5)  =MAINST
        000
922 00C7A 146         C=DAT0  A               current addr
923 00C7D 8A2         ?A=C    A               are addresses different?
924 00C80 22          GOYES   conf0           no, so continue
925                 *                          yes, so we must adjust file
926
927                 ************************************************************
928                 *
929                 * XFTH will adjust the file as needed; it will never do an
930                 * insufficient memory error; this is because FORTHRAM already
931                 * exists; someone has pulled out a number of modules and the
932                 * size of FORTHRAM must be increased, however all we are doing
933                 * is absorbing the memory freed up when the size of the configura
934                 * buffers shrunk
935                 *
936                 ************************************************************
937
938 00C82 DF          CDEX    A               save D(A) in R3
939 00C84 10B         R3=C                    I believe that R3 is not trashed
940                 *                          in any permutation of XFTH
```



```
941 00C87 22          P=      2               save 3 CPU RTN STK levels
942 00C89 8F00        GOSBVL  =R<RSTK
        000
943 00C90 7C93        GOSUB   XFTH
944 00C94 22          P=      2
945 00C96 8F00        GOSBVL  =RSTK<R         restore 3 CPU RTN STK levels
        000
946
947 00C9D 11B         C=R3
948 00CA0 D7          D=C     A               restore D(A) for config poll
949
950                 ************************************************************
951                 *
952                 * We also check the contents of oVARID.  If it contains a number
953                 * we assume that this number is the buffer ID of a buffer that
954                 * we want to save. So use I/ORES  to save buffer.
955                 *
956                 ************************************************************
957 00CA2       conf0
958 00CA2 1F34        D1=(5)  =oVARID         get buffer ID
        CF2
959 00CA9 147         C=DAT1  A               read buffer contents
960 00CAC 8AA         ?C=0    A               any to save?
961 00CAF 90          GOYES   conf2           no
962
963 00CB1 8F00        GOSBVL  =I/ORES         claim our buffer
        000
964                 *                          I/ORES uses A,C,D1,Carry
965
966 00CB8 00    conf2   RTNSXM
```

```
967                     EJECT
968                 ********************************************
969                 *
970                 * IF FORTH IS ACTIVE, WE HAVE TO DO SOMETHING
971                 * HERE. WE DECIDE WHAT TO DO BASED ON THE
972                 * VALUE OF ACTIVE, AS FOLLOWS:
973                 * 0 --------   DO NOTHING, RTNSXM
974                 *  IF ACTIVE#0
975                 *  FIRST CHECK IF FROM FORTHX (oBASIC#0)
976                 *  IF SO CLEAR ACTIVE, RETURN HAVING HANDLED POLL
977                 *  ELSE
978                 * 1 --------   ASSUME INIT1 HAS HAPPENED;
979                 *              RESET POINTERS, SHOW SIGN-ON MESSAGE
980                 *              AND ABORT
981                 * 2 --------   BASICI/F/$ -- HIT A PAUSE. SET ACTIVE
982                 *              TO 1 AND "WAKEUP" (GO TO QUIT)
983                 * 3 --------   BASICX -- COLLAPSE STACK
984                 *              AND RESTORE FORTH POINTERS; KILL
985                 *              IOBUFFER USED FOR BASIC COMMAND, CLEAN
986                 *              UP ITEMS ON RTN STACK, EXIT BASICX.
987                 * ONE OTHER WAY WE CAN GET HERE, WHICH IS NOT DEPENDENT
988                 * ON ACTIVE, IS THROUGH A MEMORY ERROR: IN THAT CASE
989                 * A GOSUB TO MFERR NEVER RETURNS BUT GOES TO MAINLOOP.
990                 * MEMERR'S ARE FLAGGED BY oERR=1 IN THE ERROR-POLL-HANDLER.
991                 * SO WHEN oERR=1 WE WANT TO ABORT.
992                 *
993                 ********************************************
994 00CBA 1BC0 abqq    D0=(5)  =ACTIVE
          BF2
995 00CC1 CE          C=C-1   A           C WAS 2, MAKE IT 1
996 00CC3 144         DAT0=C  A
997 00CC6 71F2        GOSUB   RSFP
998 00CCA 7576        GOSUB   SETI
999 00CCE 0000        CON(5)  =WAKE
          0
1000
1001 00CD3 144  abq    DAT0=C  A           RESET ERR? TO ZERO
1002 00CD6 6D82        GOTO    abortq
1003
1004
1005 00CDA 7461 BASPOL GOSUB   FTHRM?      FORTHRAM?
1006 00CDE 59D         GONC    conf2       NO JUST RTNSXM
1007
1008 00CE1 7C81        GOSUB   GOOD?       FORTHRAM in position?
1009 00CE5 52D         GONC    conf2       no just RTNSXM
1010
1011 00CE8 1BC0        D0=(5)  =ACTIVE     MAKE SURE FORTH IS
          BF2
1012 00CEF 142         A=DAT0  A           ACTIVE
1013 00CF2 8AC         ?A#0    A           ACTIVE?
1014 00CF5 50          GOYES   BPL001      YES
1015 00CF7 50C         GONC    conf2       ELSE JUST RTNSXM
1016
1017 00CFA 1B2A BPL001 D0=(5)  =oBASIC     CHECK IF FROM FORTHX
          BF2
```



```
1018 00D01 146         C=DAT0  A
1019 00D04 8AA         ?C=0    A           IF NOT,
1020 00D07 02          GOYES   BPL002              CONTINUE
1021
1022            * we're coming from FORTHX--we know this because
1023            * oBASIC contains a return addr instead of 0--however,
1024            * we may be in the middle of an implicit BASICF
1025            * i.e., FORTHX' 1.0*FNF' would invoke BASICF and would
1026            * also invoke the BASIC interpreter inorder to evaluate
1027            * the user defined function, FNF.  Therefore, we must
1028            * check to see if this is a BASICF by examining the
1029            * value in ACTIVE, if it is 2 then don't reset anything.
1030
1031            *                       A(A)=ACTIVE
1032 00D09 CC          A=A-1   A           if ACTIVE = 2 then  1
1033 00D0B CC          A=A-1   A           now 0
1034 00D0D CC          A=A-1   A           we'll carry after 3nd subtract
1035 00D0F 411         GOC     BPDONE      clear nothing, just return
1036 00D12 D2          C=0     A           set C(A)=0 and clear oBASIC & ACTIVE
1037 00D14 144         DAT0=C  A           clear oBASIC
1038 00D17 1BC0        D0=(5)  =ACTIVE
          BF2
1039 00D1E 144         DAT0=C  A           clear oBASIC
1040 00D21 00  BPDONE  RTNSXM              BACK TO MAIN LOOP
1041
1042 00D23 669F abqqq  GOTO    abqq        intermediate jump
1043
1044 00D27 3436 BPL002 LC(5)   99          HERE FROM OFF KEY FROM FORTH?
          000
1045 00D2E 8A2         ?A=C    A
1046 00D31 B7          GOYES   forpon      YES, RESTART
1047 00D33 1BBB        D0=(5)  =oERR?      ERR?=1 IF WE'RE IN MEMERR
          BF2
1048 00D3A 146         C=DAT0  A           WHICH RETURNS TO MAINLOOP
1049 00D3D CE          C=C-1   A
1050 00D3F 8AA         ?C=0    A
1051 00D42 19          GOYES   abq         abort
1052 00D44 3420        LC(5)   2           ACTIVE=2 means BASICI,F,$
          000
1053 00D4B 8A2         ?A=C    A
1054 00D4E 5D          GOYES   abqqq
1055 00D50 E6          C=C+1   A           check ACTIVE=3
1056 00D52 8A6         ?A#C    A            IF ACTIVE#3 THEN THIS MUST BE
1057 00D55 96          GOYES   init1          could be INIT1
1058 00D57 8F00        GOSBVL  =COLLAP        DON'T LEAVE MITH STK GARBAGE
          000
1059 00D5E 8F00        GOSBVL  =GETFP         RESTORE FORTH POINTERS
          000
1060
1061            *
1062            * FORTH RTN STACK HAS 3 THINGS SAVED THERE BEFORE CONTROL
1063            * PASSED TO BASIC SYSTEM: FIRST IS BUFFER ID#
1064            * OF IOBUFFER USED BY BASICX -- SAVED SO WE
1065            * CAN GET RID OF THAT BUFFER NOW;
1066            * THEN POP 2 RTNS TO CPU (SAVED FROM OS).
```

```
1067            *
1068 00D65 DC          ABEX    A              RTN STK TO A(A)
1069 00D67 132         ADOEX
1070 00D6A 146         C=DAT0  A             GET BUFFER ID TO DEALLOCATE
1071 00D6D 109         R1=C                  SAVE
1072 00D70 164         D0=D0+  5
1073 00D73 146         C=DAT0  A              GET 1ST ADDR
1074 00D76 06          RSTK=C                 PUSH BACK TO CPU STACK
1075 00D78 164         D0=D0+  5
1076 00D7B 146         C=DAT0  A              GET 2ND ADDR
1077 00D7E 06          RSTK=C                 PUSH BACK TO CPU STACK
1078 00D80 164         D0=D0+  5
1079 00D83 132         ADOEX                  NEW RTN STACK
1080 00D86 DC          ABEX    A              INTO B(A)
1081 00D88 8F00        GOSBVL  =SAVEFP        IODAL WILL USE ALL PTRS
           000
1082 00D8F 119         C=R1                   BUFFER ID FOR IODAL
1083 00D92 8F00        GOSBVL  =I/ODAL
           000
1084 00D99 84D         ST=0    13              THERE IS NO LEGITIMATE WAY FOR
1085            *                              PRGM TO BE RUNNING HERE, ALTHOUGH
1086            *                              SOMETIMES MAINLOOP HASN'T CLEARED
1087            *                              S13 YET
1088 00D9C 8F00        GOSBVL  =SVSTAT        SAVE PRGM STATUS AND CLEAR ANNUNCI
           000
1089 00DA3 8F00        GOSBVL  =GETFP
           000
1090 00DAA 03          RTNCC
1091
1092 00DAC D2  forpon  C=0     A
1093 00DAE CE          C=C-1   A              SET ACTIVE FOR RESTART
1094 00DB0 1BC0        D0=(5)  =ACTIVE
           BF2
1095 00DB7 144         DAT0=C A
1096 00DBA 6A22        GOTO    XFORTH
1097
1098            *
1099            * note: this sign on message is duplicated at
1100            *       OUTER INTERPRETER, if this message
1101            *       changes, it must also be changed there.
1102            *
1103
1104 00DBE 1F6B init1  D1=(5)  =oONERR          clear ONERR
           BF2
1105 00DC5 D2          C=0     A
1106 00DC7 145         DAT1=C  A
1107 00DCA 7DE1        GOSUB   RSFP            set D1 and B(A)
1108 00DCE 7175        GOSUB   SETI
1109 00DD2 0000        CON(5)  =PDOTQ
           0
1110 00DD7 E0          CON(2)  14
1111 00DD9 8405        NIBASC  \HP-71 FO\
           D273
           1302
           64F4
```



```
1112 00DE9 2545        NIBASC  \RTH 1A\
           8402
           1314
1113 00DF5 0000        CON(5)  =ABORT
           0
1114
1115
```

```
1116                    STITLE  FORTH_INITIALIZATION
1117              ****************************************
1118              *
1119              * ROUTINE TO CHECK THAT FILE TYPE
1120              * IN A[A] IS IN RANGE FthFil TO FthFil+3
1121              * EXIT: CARRY SET IF OK. A STILL HAS TYPE
1122              *
1123              ****************************************
1124
1125 00DFA 3491 TYPCHK  LC(5)   (=FthFil)+1
          2E0
1126              *
1127              * THERE ARE 2 FORTH TYPES, FTHFIL & FTHFIL+1 (SECURE)
1128              *
1129 00E01 E2           C=C-A  A            C=OFFSET
1130              * OFFSET MUST BE ZERO OR ONE
1131 00E03 CE           C=C-1  A            CARRY IF ZERO
1132 00E05 490          GOC    CHKYES
1133 00E08 CE           C=C-1  A            NOW CARRY IF 1
1134 00E0A 440          GOC    CHKYES
1135 00E0D 03           RTNCC
1136 00E0F 02     CHKYES RTNSC
1137
1138              * CHECK TYPE OF FILE -- WHEN D0 POINTS TO
1139              * BEGINNING OF FILE
1140              * GET TYPE IN A[A] AND CALL TYPCHK
1141              *
1142              ************************************************
1143
1144 00E11 16F  CHKTYP  D0=D0+  16             POINT PAST NAME
1145 00E14 D0           A=0    A
1146 00E16 15A3         A=DAT0 4
1147 00E1A 6FDF         GOTO   TYPCHK
1148              *                           SAME EXIT CONDITIONS
1149              *                           AS TYPCHK
1150
```



```
1151                    EJECT
1152              ************************************
1153              *
1154              * CATALOG poll handler for FORTH
1155              * type files
1156              *
1157              ************************************
1158
1159 00E1E 78DF CATPOL GOSUB  TYPCHK
1160 00E22 440          GOC    CATP0
1161 00E25 00           RTNSXM
1162
1163              * IT IS ONE OF OUR TYPES.
1164              * TYPE IS IN A[A]
1165              * SET D1 TO TYPE TABLE, A[S] TO OFFSET,A[A] TO TYPE
1166 00E27 821  CATP0  XM=0
1167 00E2A 1F00         D1=(5)  =TYTABF
          07E
1168 00E31 3471         LC(5)   (=FthFil)-1     FIRST TYPE IN TABLE
          2E0
1169              *                           MINUS ONE BECAUSE THEY WANT
1170              *                            A ONE FOR 1ST TYPE,2 FOR 2ND
1171 00E38 EE           C=A-C   A              GET "OFFSET" INTO TABLE
1172              *                           (ONE IF NORMAL TYPE)
1173 00E3A 816          CSRC                    PUT IN C[S]
1174              *                           A[A] STILL HAS TYPE
1175 00E3D ACA          A=C     S               WITH OFFSET IN S
1176 00E40 03           RTNCC
```

```
1177                EJECT
1178              ************************************************
1179              *
1180              * THIS RTNE MAKES SURE THAT:
1181              *
1182              *       1) THE FIRST FILE IN THE CHAIN (POINTED
1183              *          TO BY MAINST IS CALLED "FORTHRAM"
1184              *
1185              *       2) THAT IF THIS FILE IS CALLED
1186              *          "FORTHRAM" IT IS OF THE CORRECT
1187              *          TYPE.
1188              *
1189              * IT RETURNS WITH CARRY set IF ITS THE RIGHT FILE
1190              * AND IT RETURNS WITH CARRY clear OTHERWISE
1191              *
1192              * It assumes that if the above 2 conditions are
1193              * met that the data in FORTHRAM is in the correct
1194              * location because the configuration poll handler
1195              * will make sure of it.
1196              *
1197              ************************************************
1198
1199 00E42 20    =FTHRM? P=      0
1200 00E44 1B00          D0=(5)  =MAINST         first let's see if
          000
1201 00E4B 146           C=DAT0  A               FORTHRAM has been created!
1202 00E4E 134           D0=C                    if not, don't handle
1203 00E51 1527          A=DAT0  W               read name of 1st file in chain
1204 00E55 3F64          LCASC   \MARHTROF\      see if its FORTHRAM
          F425
          4584
          2514
          D4
1205 00E67 976           ?A#C    W               RIGHT NAME?
1206 00E6A 50            GOYES   NOFTRM          NO!
1207              *                              NOW CHECK TYPE
1208 00E6C 54A           GONC    CHKTYP          CARRY SET IF OK, ELSE CLEAR
1209
1210 00E6F 03    NOFTRM  RTNCC
1211
1212              * this routine is a further check used by Main Loop poll
1213              * handler, error poll handler and insufficient mem poll
1214              * handler, it is called after FTHRM? in these routines.
1215              * It assumes that D0 points beyond the name field in the
1216              * file header and that it can use D0, A and C.  It makes
1217              * sure that the FORTHRAM file is correctly positioned;
1218              * if it is not correctly positioned it rtns carry clear
1219              * and none of these polls should activate FORTH.
1220
1221 00E71 16F   =GOOD?  D0=D0+  16
1222 00E74 164           D0=D0-  5       point to old addr of FORTHRAM
1223 00E77 142           A=DAT0  A       read old addr
1224 00E7A 1B00          D0=(5)  =MAINST
          000
1225 00E81 146           C=DAT0  A       addr of FORTHRAM
```



```
1226 00E84 8A2           ?A=C    A       file in position?
1227 00E87 00            RTNYES          yes, rtn with carry set
1228 00E89 03            RTNCC           no, rtn with carry clear
```

```
1229                EJECT
1230
1231            ***************************
1232            *                         *
1233            *   ERROR poll handler    *
1234            *                         *
1235            ***************************
1236
1237            *-------------------------*
1238            * IS FORTHRAM FILE        *--NO---->RTN
1239            * CORRECTLY POSITIONED?   *
1240            *-------------------------*
1241            *       |
1242            *       |YES
1243            *       |
1244            *-------------------------*
1245            *  ERROR ROUTINE CALLED   *--YES--->RTN
1246            *  BY EMSG (IN ABORT)?    *         (POLLING ALREADY
1247            *-------------------------*          DONE THERE)
1248            *       |
1249            *       |NO
1250            *       |
1251            *-------------------------*
1252            * ACTIVE=-2?              *--YES-->RTN (FLAG SET BELOW...
1253            *-------------------------*             POLL GENERATED WHILE STILL
1254            *       |                             IN THIS POLL,FOR TRANSLATO
1255            *       |
1256            *       |NO
1257            *       |
1258            *-------------------------*
1259            * ACTIVE=2,3,4,5?         *--NO-->-------------------------+
1260            *-------------------------*                                 |
1261            *       |                                                 |
1262            *       |YES (IN BASICX OR BASICI,F,$)                    |
1263            *       |                                                 |
1264            *-------------------------*                               |
1265            * BASIC ON-ERROR CONDITION *--YES-->RTN (LET ON-ERROR      |
1266            * IN EFFECT?              *             HAPPEN)           |
1267            *-------------------------*                               |
1268            *       |                                                 |
1269            *       |NO                                               |
1270            *       |                                                 |
1271            *-------------------------*                               |
1272            * SAVE PROGRAM STATUS     *                               |
1273            * (SO WE SHOW SUSPEND     *                               |
1274            * WHEN WE RETURN TO BASIC *                               |
1275            *-------------------------*                               |
1276            *       |                                                 |
1277            *       | <-----------------------------------------------+
1278            *       |
1279            *-------------------------------------*
1280            * SET ACTIVE TO -2 (FOR CHECK ABOVE)  *
1281            *                                     *
1282            * COLLAPSE POLL AREA                  *
1283            *-------------------------------------*
```



```
1284            *       |
1285            *       |
1286            *-------------------------*
1287            * ONERR=0? (FORTH ON-ERROR) *--YES-->*-------------------------*
1288            *-------------------------- *       * RESET FORTH POINTERS    *
1289            *       |                           * GET cfa FROM ONERR      *
1290            *       |                           * CALL SETI               *
1291            *       |                           * DO EXEC, QUIT           *
1292            *-----------------------*           *-------------------------*
1293            * CALL MFERR*           *
1294            *-----------------------*
1295            *       |
1296            *  (MEMORY ERROR MAY NOT
1297            *   RETURN, GOES TO MAIN
1298            *   LOOP -- WE HANDLE
1299            *   MEMERR POLL SPECIFICALLY)
1300            *       |
1301            *       |
1302            *-------------------------*
1303            * SET ACTIVE TO 1         *
1304            * RESET FORTH POINTERS    *
1305            * DO ABORT                *
1306            *-------------------------*
```

```
1307                    EJECT
1308 00E8B 73BF ERRHND  GOSUB   FTHRM?          IS IT RIGHT FILE?
1309 00E8F 440          GOC     ERRH00          YES! HANDLE IT
1310
1311 00E92 00   rtnhnd  RTNSXM                  OTHERWISE JUST RTN
1312
1313 00E94 79DF ERRH00  GOSUB   GOOD?           FORTHRAM in position?
1314 00E98 59F          GONC    rtnhnd          no RTNSXM
1315
1316 00E9B 1BBB         D0=(5) =oERR?           FIRST CHECK FOR FORTH ERROR
         BF2
1317 00EA2 146          C=DAT0 A
1318 00EA5 8AA          ?C=0   A                NOT FROM EMSG? (IN FT3)
1319 00EA8 40           GOYES  ERHX0            THEN GO ON
1320 00EAA 00           RTNSXM
1321 00EAC 1BC0 ERHX0   D0=(5) =ACTIVE          LET'S SEE IF FORTH IS ALIVE
         BF2
1322 00EB3 AF2          C=0    W                CLEAR FOR CSRB BELOW
1323 00EB6 146          C=DAT0 A                READ FLAG
1324 00EB9 8AE          ?C#0   A                ACTIVE?
1325 00EBC 40           GOYES  ERRH01
1326 00EBE 00           RTNSXM                  NO, just return
1327
1328            *
1329            * WE HAVE AN ERROR OR OUT OF MEMORY CONDITION
1330            * THE FORTHRAM FILE EXISTS AND FORTH IS ACTIVE
1331            * NOW WE MUST DISPLAY THE ERROR MESSAGE AND THEN
1332            * REACTIVATE FORTH
1333            *
1334
1335 00EC0 E6   ERRH01  C=C+1  A                CHECK FOR -2
1336 00EC2 E6           C=C+1  A                SET RIGHT BELOW AT ERR03
1337 00EC4 8AE          ?C#0   A                0 MEANS THIS IS THE POLL
1338 00EC7 B0           GOYES  ERR02             WE JUST SENT
1339
1340 00EC9 D2   ERRrtn  C=0    A
1341 00ECB E6           C=C+1  A                  NEED TO CLEAR CARRY
1342 00ECD 118          C=R0                    SAVE MSG#
1343 00ED0 00           RTNSXM
1344 00ED2 1B2A ERR02   D0=(5) =oBASIC      FORTHX?
         BF2
1345 00ED9 142          A=DAT0 A
1346 00EDC 8A8          ?A=0   A
1347 00EDF 90           GOYES  ERR02+       IF SO,
1348 00EE1 8F00         GOSBVL =GTS13             RESTORE PGM-RUNNING FLAG
         000
1349            *                           SO ON ERROR CAN HAPPEN
1350            *                           (C UNTOUCHED)
1351 00EE8 86D  ERR02+  ?ST=0   13          PRGM NOT RUNNING?
1352 00EEB 74           GOYES  ERR03        THEN IGNORE ON ERROR
1353 00EED 81E          CSRB                check for ACTIVE=3 (BASICX)
1354            *                           or ACTIVE=2 (BASICI,F,$)
1355 00EF0 81E          CSRB                plus two added above, =5or4
1356            *                           (we're also testing for 6&7)
1357 00EF3 CE           C=C-1  A            lose former 4's bit
```



```
1358 00EF5 8AE          ?C#0   A
1359 00EF8 A3           GOYES  ERR03        IF IN BASIC,
1360 00EFA 1B00         D0=(5)  =ERRSUB     THEN RETURN IF ON ERROR
         000
1361 00F01 146          C=DAT0 A             IS IN EFFECT
1362 00F04 8AE          ?C#0   A
1363 00F07 2C           GOYES  ERRrtn
1364 00F09 164          D0=D0+ 5
1365 00F0C 146          C=DAT0 A
1366 00F0F 8AE          ?C#0   A
1367 00F12 7B           GOYES  ERRrtn
1368            *
1369 00F14 118          C=R0                SVST WIPES OUT R0 WHICH IS IMPORTANT
1370 00F17 109          R1=C                (IT DOESN'T TOUCH R1)
1371 00F1A 3100         LC(2)  =f1SUSP      SET SUSPEND FLAG SO PROGRAM HALTS
1372 00F1E 8F00         GOSBVL =SFLAGS      WHEN WE EXIT FORTH
         000
1373 00F25 8F00         GOSBVL =SVSTAT      SAVE CURRENT PRGM STATUS (OVERWRITE
         000
1374            *                           STATUS ON ENTRY TO FORTH) & CLEAR AN
1375 00F2C 119          C=R1
1376 00F2F 108          R0=C                RESTORE R0
1377 00F32 1BC0 ERR03   D0=(5) =ACTIVE       RESET IN CASE WE FELL THROUGH HERE
         BF2
1378 00F39 D2           C=0    A
1379 00F3B CE           C=C-1  A             FLAG CHECKED ABOVE
1380 00F3D CE           C=C-1  A              -2
1381 00F3F 144          DAT0=C A             SET IN ACTIVE
1382 00F42 8F00         GOSBVL =COLLAP       COLLAPSE POLL AREA
         000
1383 00F49 1B6B         D0=(5) =oONERR
         BF2
1384 00F50 146          C=DAT0 A             CHECK ON ERROR FLAG
1385 00F53 D7           D=C    A             SAVE POSSIBLE CFA
1386 00F55 8AE          ?C#0   A
1387 00F58 A2           GOYES  ONERX
1388 00F5A 118          C=R0                 R0 HAS MSG#
1389
1390 00F5D 8F00         GOSBVL =MFERR*       GO DO THE MESSAGE
         000
1391
1392 00F64 84D  abortq  ST=0   13
1393 00F67 1BC0         D0=(5) =ACTIVE
         BF2
1394 00F6E D2           C=0    A
1395 00F70 E6           C=C+1  A             RESET ACTIVE TO 1
1396 00F72 144          DAT0=C A
1397
1398 00F75 7240         GOSUB  RSFP
1399 00F79 76C3         GOSUB  SETI          SET D0=I AND RUN
1400 00F7D 0000         CON(5) =ABORT
         0
1401
1402            ************************************************************
1403            *
```

```
1404            * We make the assumption here that we were executing some
1405            * mainframe subroutine when an error occurred over which we
1406            * no ability to stop; since we always save the FORTH system
1407            * pointers away before we call a mainframe routine we also
1408            * assume that the info returned by GETFP (see MR/FT7) is
1409            * valid.
1410            *
1411            ************************************************************
1412
1413 00F82 1BC0 =ONERX  D0=(5) =ACTIVE        must reset ACTIVE to 1
          BF2
1414 00F89 D2           C=0    A              which means FORTH is
1415 00F8B E6           C=C+1  A              running in normal mode
1416 00F8D 144          DAT0=C A
1417 00F90 1B00         D0=(5)  =ERR#
          000
1418 00F97 118          C=R0                 ERRMSG#
1419 00F9A 15C3         DAT0=C  4            STORE ERROR NUM, WHICH MFER WOULD HAV
1420 00F9E 8F00         GOSBVL =GETFP          restore FORTH pointers
          000
1421 00FA5 DB           C=D    A              GET CFA
1422 00FA7 1C4          D1=D1- 5
1423 00FAA 145          DAT1=C A              PUT CFA ON STACK
1424 00FAD 7293         GOSUB  SETI
1425 00FB1 0000         CON(5) =EXEC
          0
1426 00FB6 0000         CON(5) =QUIT
          0
1427
1428            * RESET FORTH POINTERS
1429
1430 00FBB 1B11 RSTP    D0=(5) =oS0           START OF DATA STACK
          BF2
1431 00FC2 146          C=DAT0 A
1432 00FC5 137          CD1EX                 D1 NOW DATA STK PTR
1433 00FC8 164          D0=D0+ 5              D0-->RTN STK PTR
1434 00FCB 146          C=DAT0 A
1435 00FCE DD           BCEX   A              B(A)=RTN STK PTR
1436 00FD0 01           RTN
```



```
1437                    EJECT
1438
1439 00FD2 8D00 FORTHd GOVLNG =OUTELA
          000
1440 00FD9 03   FORTHp RTNCC
1441 00FDB 7FFF        REL(5) FORTHd
          F
1442 00FE0 9FFF        REL(5) FORTHp
          F
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER

```
1443                    EJECT
1444 00FE5 7740 *XFORTH GOSUB   XFTH
1445            * HANDLED ALL FILE INITIALIZATION
1446            * NOW SET UP INTERPRETER LOOP
1447 00FE9 63F2         GOTO    RESTRT
1448
1449
1450
1451 00FED AF2  *FRCHK  C=0     W
1452 00FF0 20           P=      0
1453 00FF2 3F64         LCASC   \MARHTROF\
           F425
           4584
           2514
           D4
1454 01004 AFA          A=C     W
1455 01007 8F00         GOSBVL  =FILEF      CHECK TO SEE IF THE RAM FILE EXISTS
           000
1456            *                           FILEF only checks in MF RAM which is
1457            *                           what we want.
1458
1459 0100E 441          GOC     FRCK0       FILE DOESN'T EXIST
1460 01011 133          AD1EX                   PUT FILEPTR IN A
1461 01014 131          D1=A                    PUT BACK IN D1 FOR FILEEX
1462 01017 130          D0=A                    TO PUT IN D0 FOR CHKTYP
1463 0101A 73FD         GOSUB   CHKTYP
1464 0101E 560          GONC    typerr          FILE EXISTS BUT IS WRONG TYPE
1465 01021 03           RTNCC
1466
1467 01023 02   FRCK0   RTNSC
1468
1469 01025 3100 typerr  LC(2)   =eFTYPE
1470 01029 8D00         GOVLNG  =MFERR
           000
1471
```



```
1472                    EJECT
1473 01030 79BF *XFTH   GOSUB   FRCHK       DOES FILE EXIST?
1474 01034 460          GOC     INIT05      NO
1475 01037 6B13         GOTO    FILEEX      YES, IT DOES
1476
1477            ********************************************************
1478            * WE MAY NEED THIS ADDR FOR CUSTOM ROMS (SOFT ADDRESSED)
1479            ********************************************************
1480            * Determine ROM base addr, return it in C(A)
1481            *
1482            *
1483            * *ROMADR GOSUB  ROMST       determine start addr of this ROM
1484            *  ROMST  C=RSTK
1485            *         C=0     X          clear offset from start of file
1486            *         RTN                C has ROM start addr
1487
1488            *-------------------------------------------------------------
1489            *
1490            * OUR RAM FILE DOES NOT EXIST, THEREFORE, WE MUST BLOW OPEN
1491            * A HOLE AT THE START OF THE FILE CHAIN AND CREATE THE
1492            * SUCKER.
1493            *
1494
1495 0103B 349A  INIT05 LC(5)   FSIZET          FILE SIZE APPROX 3K BYTES
           610
1496 01042 D5           B=C     A               FILE SIZE IN NIBS
1497 01044 343D         LC(5)   =MINFIL         ADJUSTIABLE FILE START
           AF2
1498            *
1499 0104B DA           A=C     A
1500 0104D 1B00         D0=(5)  =MAINST         POINTER TO START OF CHAIN
           000
1501 01054 146          C=DAT0  A               C(A)=START OF 1ST FILE
1502 01057 EE           C=A-C   A               NEEDED DEAD SPACE IN FILE
1503
1504 01059 412          GOC     BIGERR          IF THIS NUMBER IS NEGATIVE
1505            *                               THEN WE HAVE CONFIGURATION
1506            *                               PROBLEM--FORTHRAM WILL NOT BE
1507            *                               ABLE TO BE PLACED CORRECTLY
1508            *
1509
1510 0105C C1           B=B+C   A               REAL NECESSARY SIZE TO CREATE
1511 0105E D9           C=B     A
1512 01060 109          R1=C                    SAVE COPY IN R1 FOR LATER
1513 01063 146          C=DAT0  A               START OF 1ST FILE
1514 01066 DA           A=C     A               A(A)=START OF 1ST FILE
1515 01068 8F00         GOSBVL  =MVMEM+         A(A)=C(A)=1ST FILE IN CHAIN,
           000
1516            *                               P=0, B(A)=OFFSET(FILE SIZE)
1517
1518 0106F 5E1          GONC    INIT10          FILE CREATED
1519 01072 20   ERR     P=      0               NOT CREATED, ERROR
1520 01074 8D00         GOVLNG  =MFERR          JUMP TO ERROR CODE
           000
1521
```

```
1522 0107B 22   BIGERR  P=      2
1523 0107D 31F2         LC(2)   =ID
1524 01081 20           P=      0
1525 01083 3143         LC(2)   =eCONF          GIVE CONFIGURATION ERR
1526 01087 8D00         GOVLNG  =BSERR
           000
1527
1528 0108E 110  INIT10  A=R0                    A(A)=START OF FILE
1529 01091 130          D0=A
1530 01094 131          D1=A
1531 01097 3F64         LCASC   \MARMTROF\
           F425
           4584
           2514
           D4
1532 010A9 1557         DAT1=C W                WRITE OUT FILENAME
1533 010AD 17F          D1=D1+ 16
1534 010B0 3381         LC(4) =FthFil           FILE TYPE
           2E
1535 010B6 15D3         DAT1=C 4                WRITE OUT FILE TYPE
1536 010BA 17F          D1=D1+  16
1537 010BD 111          A=R1                    RECOVER FILE SIZE
1538 010C0 3402         LC(5)   32              HEADER OVERHEAD
           000
1539 010C7 EE           C=A-C   A               OFFSET TO NEXT FILE IN CHAIN
1540 010C9 145          DAT1=C  A               WRITE TO FILE
1541
1542 010CC 174          D1=D1+  5               POINT TO RAMFILE ADDR HOLDER
1543 010CF 136          CD0EX
1544 010D2 145          DAT1=C  A               WRITE OUT RAMFILE START ADDR
1545 010D5 136          CD0EX
1546
1547 010D8 8F00         GOSBVL  =WFTMDT         D0->START OF FILE; WRITE
           000
1548            *                               FLAGS, TIME & CREATION
1549            *                               DATE TO FILE
1550
1551 010DF 79B1         GOSUB   TBLMOV          move stuff to ram
1552 010E3 9B10         CON(5)  =TBLLEN         length of table
           0
1553            ****************************************
1554            *                                      *
1555            *    INITIALIZATION OF RAM FILE        *
1556            *                                      *
1557            ****************************************
1558 010E8 C7D0         CON(5)  =DSTK           DSTKAD
           3
1559 010ED C0F0         CON(5)  =RSTK           RSTKAD
           3
1560 010F2 0000         CON(5)  0               IREG
           0
1561 010F7 1000         CON(5)  1               ACTIVE
           0
1562 010FC C7D0         CON(5)  =DSTK           S0
           3
```



```
1563 01101 C0F0         CON(5)  =RSTK           R0
           3
1564 01106 C7D0         CON(5)  =TIBF           TIB
           3
1565 0110B CDF0         CON(5)  =USEIT          USE
           3
1566 01110 C0F0         CON(5)  =RSTK           PREV
           3
1567 01115 C0F0         CON(5)  =RSTK           FIRST
           3
1568 0111A C711         CON(5)  =LAST+1         LIMIT
           3
1569 0111F CACF         CON(5)  =FRTHVL         VOC-LINK
           2
1570 01124 0600         CON(5)  RECSIZ          SET RECORD SIZE TO 96 BYTES
           0
1571 01129 0000         CON(5)  0               #TIB
           0
1572 0112E F100         CON(5)  NAMLEN          WIDTH
           0
1573 01133 FFFF         CON(5)  #FFFFF          WARNING
           F
1574 01138 0000         CON(5)  0               OKFLG
           0
1575 0113D 0000         CON(5)  0               BLK
           0
1576 01142 0000         CON(5)  0               IN
           0
1577 01147 0000         CON(5)  0               SPAN
           0
1578 0114C 0000         CON(5)  0               SCR
           0
1579 01151 7ACF         CON(5)  FRTHVC          CONTEXT
           2
1580 01156 7ACF         CON(5)  FRTHVC          CURRENT
           2
1581 0115B 0000         CON(5)  0               STATE
           0
1582 01160 A000         CON(5)  10              BASE
           0
1583 01165 0000         CON(5)  0               DPL
           0
1584 0116A 0000         CON(5)  0               FLD
           0
1585 0116F 0000         CON(5)  0               CSP
           0
1586 01174 0000         CON(5)  0               HLD
           0
1587 01179 1BCF         CON(5)  DICTST          FENCE
           2
1588 0117E 1BCF         CON(5)  DICTST          DP
           2
1589 01183 0D00         CON(5)  BUFLEN          BUFFER LENGTH IN NIBS
           0
1590 01188 0000         CON(5)  0               CURRENT LINE # IN ACTIVE BUFFER
```

```
          0
1591 0118D 0000         CON(5)  0               BASIC RETURN
          0
1592 01192 1000         CON(5)  1               HPIL LOOP #
          0
1593 01197 0000         CON(5)  0               HPIL SECONDARY ADDR
          0
1594 0119C 1000         CON(5)  1               HPIL PRIMARY ADDR
          0
1595 011A1 0000         CON(5)  0               ON ERROR
          0
1596 011A6 0000         CON(5)  0               ERROR?
          0
1597              * MORE VARIABLES...
1598              * FLOATING POINT STACK TAKES 80 NIBS!
1599              * (VARIABLES oX,oY,oZ,oT)
1600 011AB              BSS   80
1601              * FILENAME VARIABLE TAKES 8 BYTES, 16 NIBS
1602 011FB              BSS   16
1603              *
1604              * INDIRECTION VECTORS
1605              *
1606 0120B 0000         CON(5)  =IINTRP         INTERPRET
          0
1607 01210 0000         CON(5)  =ICREAT         CREATE
          0
1608 01215 0000         CON(5)  =INUMB          NUMBER
          0
1609 0121A 0000         CON(5)  =ICOMMA         COMMA
          0
1610 0121F 0000         CON(5)  =ICCOMA         C,
          0
1611 01224 0000         CON(5)  =IALLOT         ALLOT
          0
1612 01229 0000         CON(5)  =IUNIQ          not unique message
          0
1613
1614              * ASSEMBLER VARIABLES
1615
1616 0122E 0000         CON(5)  0               VARID -- ASSEMBLER VARIABLE
          0
1617 01233 8300         CON(5)  56              PAGESIZE
          0
1618
1619              * NOW THE STRING VARIABLE LISTING
1620              * IT LOOKS LIKE: 20 00 {40 NIBBLES} -- i.e.,STRING OF MAXLEN 20
1621 01238 41           CON(2)  20
1622 0123A              BSS     42
1623              * 2 MORE ASSEMBLER VARIABLES
1624 01264 0000         CON(5)  0               PARRT
          0
1625 01269 0000         CON(5)  0               PRORT
          0
1626              * MORE VARIABLES
1627 0126E 0000         CON(5)  0               PSTAT -- PROGRAM STATUS SAVE
```



```
          0
1628 01273 0000         CON(5)  0               UNUSED VARIABLE AS OF 11-8-83
          0
1629              *
1630              *---------------------------------------------------
1631              * END OF VARIABLES
1632              *
1633              * FORTH - first entry in RAM dictionary
1634
1635 01278 0000         CON(5)  0               end of RAM dictionary chain
          0
1636              *FRTHHD goes here
1637 0127D 5C           CON(2)  #C5
1638 0127F 64F4         NIBASC  \FORT\
          2545
1639 01287 8C           CON(2)  \H\+#80
1640 01289 0000         CON(5)  =DOVOC
          0
1641 0128E 18           CON(2)  #81
1642 01290 0A           CON(2)  \ \+#80
1643              *FRTHVC goes here
1644 01292 29CF         CON(5)  FRTHHD
          2
1645              *FRTHVL goes here
1646 01297 0000         CON(5)  0
          0
1647              *DICTST goes here
1648
1649 0129C 07   TBLMOV  C=RSTK
1650 0129E 135          D1=C    A
1651 012A1 AF0          A=0     W               clear for shift below
1652 012A4 143          A=DAT1  A               get count
1653 012A7 174          D1=D1+  5
1654 012AA 1BDF         D0=(5)  =DSTKAD
          AF2
1655              *
1656              * enter NMOVE after MOVSET with word count in A[A]
1657              * and nibbles extra in A[S]
1658              *      D1=source address
1659              *      D0=target address
1660              *
1661 012B1 814          ASRC                    /16, REMAINDER IN A[S]
1662 012B4 8F00         GOSBVL  =NMOVE1
          000
1663              *
1664              * NOW SAVE PROGRAM STATUS, INCASE WE HAVE BEEN RUNNING
1665              * OR WANT TO TURN ANNUNCIATORS BACK ON AT THE END
1666 012BB 8F00         GOSBVL  =SVSTAT
          000
1667 012C2 20           P=      0
1668 012C4 1F11         D1=(5)  =oS0
          BF2
1669 012CB 147          C=DAT1  A               DATA STK ADDR
1670 012CE 135          D1=C                    D1=DATA STK PTR
1671 012D1 1B61         D0=(5)  =oR0
```

```
          BF2
1672 012D8 146          C=DAT0  A               SET C=RTN STK PTR
1673 012DB 01           RTN
1674
1675            *
1676            * IF CALLED FROM XFORTH, WE ESSENTIALLY DROP THROUGH HERE
1677            *
1678
1679 012DD D5   RESTRT  B=C     A               B(A)=RTN STK PTR
1680 012DF 1BC0         D0=(5)  =ACTIVE         SET FORTH AS ACTIVE
          BF2
1681 012E6 142          A=DAT0  A               SEE WHAT'S THERE NOW
1682 012E9 D2           C=0     A
1683 012EB E6           C=C+1   A
1684 012ED 144          DAT0=C  A               WRITE A ONE TO ACTIVE
1685 012F0 1B2A         D0=(5)  =oBASIC         reset oBASIC to 0
          BF2
1686 012F7 CE           C=C-1   A               C(A)=0
1687 012F9 144          DAT0=C  A
1688 012FC E4           A=A+1   A               CHECK FOR -1 LEFT BY SLEEP
1689 012FE 4E2          GOC     WAKEUP
1690
1691 01301 7E30         GOSUB   SETI            SET I TO QUIT RTNE
1692            *                               WHICH IS REALLY THE
1693            *                               OUTER INTERPRETER
1694            *
1695            * OUTER INTERPRETER
1696            *
1697            * note: this sign on message is replicated at init1
1698            *       if this message changes it must also be changed
1699            *       there!
1700            *
1701 01305 0000         CON(5)  =PDOTQ
          0
1702 0130A E0           CON(2)  14
1703 0130C 8405         NIBASC  \HP-71 FO\
          D273
          1302
          64F4
1704 0131C 2545         NIBASC  \RTH 1A\
          8402
          1314
1705 01328 0000         CON(5)  =QUIT
          0
1706
1707 0132D 1BDF WAKEUP  D0=(5)  =DSTKAD         data stack save area
          AF2
1708 01334 146          C=DAT0  A               read old data stack value
1709 01337 137          CD1EX                   restore old data stk
1710 0133A 7500         GOSUB   SETI
1711 0133E 0000         CON(5)  =WAKE
          0
1712
1713
1714 01343 07   =SETI   C=RSTK                  POP ADDR OF OI
```



```
1715 01345 136          CD0EX                   D0=I
1716 01348 8D00         GOVLNG =NEXT00
          000
1717
1718
1719            * RE-ENTERING FORTH, THE FILE ALREADY EXISTS
1720
1721 0134F 622D ERR1    GOTO    ERR             ERROR OUT
1722
1723 01353 1B00 FILEEX  D0=(5)  =MAINST
          000
1724 0135A 343D         LC(5)   =MINFIL           HIGHEST POSSIBLE START ADDR
          AF2
1725 01361 DE           ACEX    A
1726 01363 146          C=DAT0  A               C(A)=ADDR OF 1ST FILE
1727 01366 8B6          ?A>C    A                 ARE WE SAFE???
1728 01369 60           GOYES   OKY               YES
1729 0136B 6F0D         GOTO    BIGERR            NO
1730
1731 0136F 133  OKY     AD1EX                   A(A)=ADDR OF OUR FILE
1732 01372 131          D1=A
1733
1734 01375 8A2          ?A=C    A               IS FILE IN PLACE?
1735 01378 60           GOYES   OKY0            YES, SEE IF IT NEEDS ADJUSTING
1736 0137A 6680         GOTO    MVFILE          NO, MOVE IT
1737
1738 0137E 3452 OKY0    LC(5)   37              SET D0-->OLD RAM FILE ADDR
          000
1739 01385 C2           C=A+C   A               WHICH LIVES AFTER FILE
1740 01387 134          D0=C                    HEADER
1741
1742 0138A 146          C=DAT0  A               C(A)=OLD FILE ADDR
1743 0138D 140          DAT0=A  A               WRITE OUT NEW "OLD FILE ADDR"
1744
1745 01390 8A2          ?A=C    A               NO SHIFT NECESSARY?
1746 01393 83           GOYES   GOFRTH          RIGHT!
1747
1748            *
1749            * CASE 1:
1750            *
1751            * IF NEW FILE ADDR IS LOWER IN MEMORY THAN OLD FILE ADDR
1752            * THEN WE MUST INCREASE THE AMOUNT OF DEAD SPACE IN RAM FILE
1753            *
1754            * CASE 2:
1755            *
1756            * IF NEW FILE ADDR IS HIGHER IN MEMORY THAN OLD FILE ADDR
1757            * THEN WE MUST DECREASE THE AMOUNT OF DEAD SPACE IN RAM FILE
1758            *
1759
1760 01395 8B6          ?A>C    A               NEW > OLD ?
1761 01398 51           GOYES   CASE2           YES
1762
1763 0139A D5   CASE1   B=C     A
1764 0139C E8           B=B-A   A               B=(OLD-NEW), A POSITIVE #
1765 0139E 34A2         LC(5)   42              OVRHD+5 FOR RAM FILE ADDR
```

```
          000
1766 013A5 C2          C=C+A   A
1767 013A7 DE          ACEX    A              A(A)=(START OF DEAD SPACE)
1768           *                              B(A)=# NIBS TO BLOW OPEN
1769           *                              C(A)=FILE START ADDR
1770 013A9 6A10        GOTO    SHIFT
1771
1772 013AD 130 CASE2   D0=A                   D0--> FILE
1773 013B0 D5          B=C     A
1774 013B2 EE          C=A-C   A              (NEW-OLD), A POSITIVE #
1775 013B4 CA          A=A+C   A              FILE START + (NEW-OLD)
1776 013B6 34A2        LC(5)   42
          000
1777 013BD CA          A=A+C   A              A(A)=DEAD SPACE + (NEW-OLD)
1778 013BF 136         CD0EX                  C(A)=NEW FILE ADDR
1779 013C2 E1          B=B-C   A              B(A)=(OLD-NEW), NEG #
1780
1781 013C4 8F00 SHIFT  GOSBVL  =MVMEM+        SHIFT FILE
          000
1782
1783 013CB 1B6B GOFRTH D0=(5)  =oONERR        ZERO ON ERROR FLAG
          BF2
1784 013D2 D2          C=0     A              (THIS IS FOR START-UP
1785 013D4 144         DAT0=C  A                FILE-EXISTS PATH OF BOTH
1786           *                                FORTH AND XFTH)
1787 013D7 1BBB        D0=(5)  =oERR?         FLAG USED BY EMSG, CHECKED IN BYE
          BF2
1788 013DE 144         DAT0=C  A              CLEAR IT
1789 013E1 8F00        GOSBVL  =SVSTAT          save BASIC program  status
          000
1790 013E8 1BDF        D0=(5)  =DSTKAD          TOS SAVED BY BYE
          AF2
1791 013EF 146         C=DAT0  A                READ START OF DATA STACK
1792 013F2 135         D1=C                     SET D1 UP AS DATA STACK
1793 013F5 1B61        D0=(5)  =oR0
          BF2
1794 013FC 146         C=DAT0  A                READ START OF RTN STK
1795 013FF 01          RTN                    (SET UP FOR RESTRT)
1796
1797           *
1798           * THIS AMBITIOUS ROUTINE WILL ADJUST THE FILE ORDER TO
1799           * MAKE SURE THAT THE FORTHRAM FILE IS FIRST IN THE CHAIN.
1800           * AS LONG AS WE HAVE 37 BYTES OF FREE MEMORY WE CAN SWITCH
1801           * FILE ORDER.
1802           *
1803
1804           *
1805           * FIND AVM (AVAILABLE MEMORY); ERROR OUT IF AVM < 37 BYTES
1806           * (BOTH OLD AND NEW FILES MUST KEEP THEIR HEADERS DURING THIS
1807           *  AND EACH HEADER REQUIRES 37 NIBS)
1808           *
1809
1810 01401 17F MVFILE  D1=D1+  16
1811 01404 17F         D1=D1+  16              PT TO LINK FIELD OF OLD FILE
1812 01407 137         CD1EX
```



```
1813 0140A 134         D0=C                   D0 IS NOT TRASHED BY MEMCKL
1814
1815 0140D 20          P=      0
1816 0140F 34A4        LC(5)   74             MIN NIBS TO DO TRANSFER
          000
1817           *
1818           * IT WOULD BE HANDY TO NOT HAVE TO ALLOW FOR LEEWAY'S WORTH
1819           * OF MEMORY HERE, SIMPLY BY SETTING P=1, HOWEVER....
1820           * MVMEM+ DOES INSIST THAT THERE BE ENOUGH MEMORY TO
1821           * ACCOMMODATE LEEWAY PLUS THE AMOUNT OF THE MOVE.
1822           *
1823
1824 01416 8F00        GOSBVL  =MEMCKL        CHECK FREE MEMORY
          000
1825 0141D 560         GONC    MVOK
1826 01420 6E2F ERR2   GOTO    ERR1           NOT ENOUGH MEMORY
1827
1828 01424 DE  MVOK    ACEX    A              A(A)=AVM-74
1829 01426 C0          A=A+B   A              ADD 74 BACK IN
1830 01428 136         CD0EX                  SAVE LINK FIELD OF OLD
1831 0142B 06          RSTK=C                 FILE ON CPU RTN STK
1832 0142D 134         D0=C
1833
1834           *
1835           * TEST TO SEE IF AVM => FILE SIZE; IF SO MOVE WHOLE FILE, ELSE
1836           * MOVE AVM
1837           *
1838
1839 01430 3402        LC(5)   32             # NIBS OVERHD PRECEEDING LINK FIELD
          000
1840 01437 DD          BCEX    A              INTO B(A)
1841
1842 01439 146         C=DAT0  A              # NIBS TO NEXT FILE
1843 0143C C9          C=C+B   A              TOTAL FILE SIZE
1844
1845 0143E 8B2         ?A<C    A              AVM < REQUIRED ? A=AVM
1846 01441 40          GOYES   NEMEM          NOT ENOUGH MEMORY TO TRANSFER WHOLE
1847           *                              FILE, TRANSFER AVM
1848
1849 01443 DA          A=C     A              PLENTY OF MEMORY, TRANSFER FILE
1850
1851           *
1852           * OPEN AVM AT @MAINST FOR NEW FILE
1853           *
1854
1855 01445 DC  NEMEM   ABEX    A              B(A)=#NIBS TO OPEN
1856 01447 1F00        D1=(5)  =MAINST        SET BOTH A(A) & C(A)
          000
1857 0144E 147         C=DAT1  A              TO @MAINST, THIS TELLS MVMEM+
1858 01451 DA          A=C     A              WE ARE CREATING A NEW FILE
1859
1860 01453 8F00        GOSBVL  =MVMEM+        B(A)=AVM STILL
          000
1861 0145A 45C         GOC     ERR2           out of memory!!!!
1862
```

```
1863            *
1864            * MOVE AVM WORTH OF OLD FILE TO NEW FILE
1865            *
1866
1867 0145D 07          C=RSTK                  C=LINK FIELD OF OLD FILE
1868 0145F C9          C=C+B   A               MUST ADJUST OLD ADDR BY AVM
1869 01461 06          RSTK=C
1870
1871 01463 136         CD0EX                   B(A)=AVM; COUNT
1872 01466 18F         D0=D0-  16
1873 01469 18F         D0=D0-  16
1874 0146C 132         AD0EX                   A(A)=ADDR OF OLD FILE; SOURCE
1875 0146F 11A         C=R2                    HEADER OF NEW FILE STORED
1876            *                              HERE BY MVMEM+; DESTINATION
1877
1878 01472 8F00        GOSBVL  =MOVE*M         MOVE BLOCK OF MEMORY
         000
1879            *                              A,B,C = ENTRY CONDITIONS
1880
1881            *
1882            * SET LINK FIELD IN NEW FILE
1883            *
1884
1885 01479 134         D0=C
1886 0147C 16F         D0=D0+  16
1887 0147F 16F         D0=D0+  16              D0--> LINK FIELD OF NEW FILE
1888
1889 01482 340E        LC(5)   (0-32)          SET LINK TO AVM-32
         FFF
1890 01489 C9          C=C+B   A
1891 0148B 144         DAT0=C  A
1892
1893            *
1894            * RECOVER AVM-37 NIBS OF MEMORY; IF NOT DONE
1895            * AVM=AVM-37
1896            *
1897            * STATUS: A(A)=ADDR OF OLD FILE; P=0; B(A)=AVM
1898            *
1899
1900 0148E 130         D0=A
1901 01491 C0          A=A+B   A               A(A)=(OLD FILE START+AVM)
1902
1903 01493 3452        LC(5)   37              RECOVER ABS|37-AVM| NIBS
         000
1904 0149A E9          C=C-B   A               FROM OLD FILE
1905 0149C DD          BCEX    A
1906
1907 0149E 06          RSTK=C                  SAVE AVM ON CPU STACK
1908
1909 014A0 136         CD0EX                   C(A)=OLD FILE HEADER ADDR
1910 014A3 8F00        GOSBVL  =MVMEM+         RECOVER MEMORY
         000
1911
1912            *
1913            * TEST TO SEE IF WE'RE DONE (LINK FIELD OF OLD FILE <=5
```



```
1914            *
1915
1916 014AA 3452        LC(5)   37              AFTER MOVING HEADER WE
         000
1917 014B1 DA          A=C     A               37 NIBS LESS OF AVM TO PLAY WITH
1918 014B3 07          C=RSTK
1919 014B5 E2          C=C-A   A
1920
1921 014B7 D5  AGAIN   B=C     A               B(A)= NEW AVM
1922 014B9 07          C=RSTK
1923 014BB 06          RSTK=C
1924 014BD 134         D0=C                    D0=LINK FIELD OF OLD FILE
1925
1926 014C0 3450        LC(5)   5
         000
1927 014C7 142         A=DAT0  A               A(A)=LINK
1928 014CA 8B6         ?A>C    A               MORE TO DO?
1929 014CD 52          GOYES   NDONE           NOT DONE YET
1930
1931            *
1932            * DONE: NOW GET RID OF OLD FILE'S HEADER
1933            *
1934
1935 014CF 34BD        LC(5)   #FFFDB          (-37 NIBS) TO RECOVER
         FFF
1936 014D6 DD          BCEX    A               B(A)=AMT TO RECOVER
1937 014D8 07          C=RSTK                  C(A)=LINK FIELD OF OLD FILE
1938 014DA DE          ACEX    A               CLEAN UP CPU RTN STK
1939 014DC 3450        LC(5)   5
         000
1940 014E3 CA          A=A+C   A               SET BOTH A&C TO PT TO NEXT
1941 014E5 D6          C=A     A               FILE IN CHAIN
1942 014E7 8F00        GOSBVL  =MVMEM+         DELETE OLD FILE HEADER
         000
1943
1944            *
1945            * NOW WE'RE ALL DONE & WE WANT TO CLEAN UP THE DEADSPACE
1946            * IN THE FILE SO JUMP BACK TO THE START OF THIS SECTION
1947            *
1948 014EE 614B        GOTO    XFTH
1949
1950            *
1951            * WE'RE NOT DONE & WE HAVE TO MOVE SOME MORE OF THE OLD
1952            * FILE.
1953            *
1954            * STATUS: B(A)=AVM;  RSTK--> OLD FILE LINK ADDR
1955            *         A(A)=OLD FILE LINK   C(A)=5
1956            *         D0-->OLD LINK FIELD
1957
1958 014F2 EA    NDONE   A=A-C   A               A(A)=#NIBS LEFT TO MOVE
1959 014F4 8B0         ?B<A    A               AVM < NEEDED SPACE?
1960 014F7 40          GOYES   NEMEM2          CAN ONLY MOVE AVM NIBS
1961 014F9 D8          B=A     A               PLENTY OF MEM, ONLY MOVE
1962            *                              NEEDED NIBS
1963
```

```
1964            *
1965            * OPEN AVM'S WORTH OF NIBS AT END OF NEW FILE
1966            *
1967
1968 014FB 1F00 NEMEM2  D1=(5)  =MAINST       C(A)=START OF NEW FILE
          000
1969 01502 147          C=DAT1  A
1970 01505 135          D1=C
1971 01508 17F          D1=D1+  16
1972 0150B 17F          D1=D1+  16
1973 0150E 143          A=DAT1  A             A(A)=LINK FIELD OF NEW FILE
1974 01511 137          CD1EX
1975 01514 CA           A=A+C   A             ADDR OF NEW FILE; SOURCE FOR MOVE
1976 01516 137          CD1EX                 C(A)=START OF NEW FILE
1977 01519 8F00         GOSBVL  =MVMEM+       OPEN HOLE AT END OF NEW FILE
          000
1978            *                             THIS TIME MVMEM+ ADJUSTS THE LINK
1979            *                             FIELD OF NEW FILE
1980
1981 01520 07           C=RSTK
1982 01522 C9           C=C+B   A             AJUST OLD LINK FIELD BY AVM
1983 01524 06           RSTK=C
1984
1985 01526 DE           ACEX    A
1986 01528 3450         LC(5)   5
          000
1987 0152F CA           A=A+C   A             A(A)=SOURCE OF MEM TO MOVE
1988 01531 118          C=R0                  C(A)=DESTINATION
1989
1990 01534 8F00         GOSBVL  =MOVE*M       EXIT: A(A)=SOURCE
          000
1991            *                                   B(A)=AVM
1992            *                                   C(A)=DESTINATION
1993
1994            *
1995            * COLLAPSE OLD FILE BY AVM
1996            *
1997
1998 0153B 3452         LC(5)   37            C(A)=OLD FILE HEADER
          000
1999 01542 EE           C=A-C   A
2000 01544 C0           A=A+B   A             A(A)=SOURCE+AVM
2001 01546 134          D0=C
2002 01549 D2           C=0     A
2003 0154B DD           BCEX    A
2004 0154D 06           RSTK=C                SAVE AVM ON STACK
2005 0154F E1           B=B-C   A             B(A)=0-AVM
2006 01551 136          CD0EX                 C(A)=OLD FILE HEADER
2007
2008 01554 8F00         GOSBVL  =MVMEM+
          000
2009
2010 0155B 07           C=RSTK                C(A)=AVM
2011 0155D 695F         GOTO    AGAIN
2012 01561              END
```



```
ABORT   Ext                -  1113  1400
AGAIN   Rel   5303 #014B7 -  1921  2011
BASPOL  Rel   3290 #00CDA -  1005   869
BIGERR  Rel   4219 #0107B -  1522  1504  1729
BPDONE  Rel   3361 #00D21 -  1040  1035
BPL001  Rel   3322 #00CFA -  1017  1014
BPL002  Rel   3367 #00D27 -  1044  1020
BSERR   Ext                -  1526
CASE1   Rel   5018 #0139A -  1763
CASE2   Rel   5037 #013AD -  1772  1761
CATP0   Rel   3623 #00E27 -  1166  1160
CATPOL  Rel   3614 #00E1E -  1159   870
CHKTYP  Rel   3601 #00E11 -  1144  1208  1463
CHKYES  Rel   3599 #00E0F -  1136  1132  1134
COLLAP  Ext                -  1058  1382
DOVOC   Ext                -  1640
ERHX0   Rel   3756 #00EAC -  1321   885  1319
ERR     Rel   4210 #01072 -  1519  1721
ERR#    Ext                -  1417
ERR02   Rel   3794 #00ED2 -  1344  1338
ERR02+  Rel   3816 #00EE8 -  1351  1347
ERR03   Rel   3890 #00F32 -  1377  1352  1359
ERR1    Rel   4943 #0134F -  1721  1826
ERR2    Rel   5152 #01420 -  1826  1861
ERRH00  Rel   3732 #00E94 -  1313  1309
ERRH01  Rel   3776 #00EC0 -  1335  1325
ERRHND  Rel   3723 #00E8B -  1308   871
ERRSUB  Ext                -  1360
ERRrtn  Rel   3785 #00EC9 -  1340  1363  1367
EXEC    Ext                -  1425
FILEEX  Rel   4947 #01353 -  1723  1475
FILEF   Ext                -  1455
FILEND  Ext                -    23
FORTH$  Ext                -    56
FORTHF  Ext                -    51
FORTHI  Ext                -    47
FORTHX  Ext                -    43
FORTHd  Rel   4050 #00FD2 -  1439  1441
FORTHp  Rel   4057 #00FD9 -  1440  1442
=FROST  Rel      0 #00000 -    16
=FRCHK  Rel   4077 #00FED -  1451  1473
 FRCKO  Rel   4131 #01023 -  1467  1459
=FTHRM?  Rel  3650 #00E42 -  1199   877   906  1005  1308
 GETFP  Ext                -  1059  1089  1420
 GOFRTH  Rel  5067 #013CB -  1783  1746
=GOOD?  Rel   3697 #00E71 -  1221   879  1008  1313
 GTS13  Ext                -  1348
 I/ODAL  Ext               -  1083
 I/ORES  Ext               -   963
 IALLOT  Ext               -  1611
 ICCOMA  Ext               -  1610
 ICOMMA  Ext               -  1609
 ICREAT  Ext               -  1607
 IINTRP  Ext               -  1606
 INIT05  Rel   4155 #0103B -  1495  1474
```

```
 INIT10  Rel   4238 #0108E -  1528  1518
 INUMB   Ext              -  1608
 IUNIQ   Ext              -  1612
 MAINST  Ext              -   921  1200  1224  1500  1723  1856  1968
 MEMCKL  Ext              -  1824
 MFERR   Ext              -  1470  1520
 MFERR*  Ext              -  1390
 MOVE*M  Ext              -  1878  1990
 MVFILE  Rel   5121 #01401 -  1810  1736
 MVMEM+  Ext              -  1515  1781  1860  1910  1942  1977  2008
 MVOK    Rel   5156 #01424 -  1828  1825
 MsgTbl  Rel    191 #000BF -   120    32
 NDONE   Rel   5362 #014F2 -  1958  1929
 NEMEM   Rel   5189 #01445 -  1855  1846
 NEMEM2  Rel   5371 #014FB -  1968  1960
 NEXT00  Ext              -  1716
 NMOVE1  Ext              -  1662
 NOFTRM  Rel   3695 #00E6F -  1210  1206
 OKY     Rel   4975 #0136F -  1731  1728
 OKY0    Rel   4990 #0137E -  1738  1735
=ONERX   Rel   3970 #00F82 -  1413  1387
 OUTELA  Ext              -  1439
 PDOTQ   Ext              -  1109  1701
 POLHND  Rel   3022 #00BCE -   834    33
 QUIT    Ext              -  1426  1705
 R<RSTK  Ext              -   942
 RESTRT  Rel   4829 #012DD -  1679  1447
 RSFP    Rel   4027 #00FBB -  1430   997  1107  1398
 RSTK<R  Ext              -   945
 SAVEFP  Ext              -  1081
=SETI    Rel   4931 #01343 -  1714   998  1108  1399  1424  1691  1710
 SFLAGS  Ext              -  1372
 SHIFT   Rel   5060 #013C4 -  1781  1770
 SVSTAT  Ext              -  1088  1373  1666  1789
 TBLMOV  Rel   4764 #0129C -  1649  1551
 TYPCHK  Rel   3578 #00DFA -  1125  1147  1159
 TxTbEn  Rel    188 #000BC -    86
 TxTbSt  Rel    115 #00073 -    64    31
 VER$en  Rel   3121 #00C31 -   865   865   858
 VER$st  Rel   3105 #00C21 -   864   865   858
 WAKE    Ext              -   999  1711
 WAKEUP  Rel   4909 #0132D -  1707  1689
 WFTMDT  Ext              -  1547
=XFORTH  Rel   4069 #00FE5 -  1444    39  1096
=XFTH    Rel   4144 #01030 -  1473   943  1444  1948
 abortq  Rel   3940 #00F64 -  1392  1002
 abq     Rel   3283 #00CD3 -  1001  1051
 abqq    Rel   3258 #00CBA -   994  1042
 abqqq   Rel   3363 #00D23 -  1042  1054
 baspol  Rel   3127 #00C37 -   869   840
 catpol  Rel   3131 #00C3B -   870   843
 conf0   Rel   3234 #00CA2 -   957   924
 conf2   Rel   3256 #00CB8 -   966   961  1006  1009  1015
 config  Rel   3171 #00C63 -   906   852
 eFEXST  Ext              -   611
```



```
 eFSPEC  Ext              -   332
 eFTYPE  Ext              -  1469
 eNFOUN  Ext              -   161
 errMEM  Rel   3139 #00C43 -   877   849
 errhnd  Rel   3135 #00C3F -   871   846
 fLEX    Ext              -    19
 flSUSP  Ext              -  1371
 forpon  Rel   3500 #00DAC -  1092  1046
 hVER$0  Rel   3079 #00C07 -   855   837
 hVER$1  Rel   3125 #00C35 -   866   861
 hVER$2  Rel   3125 #00C35 -   867   854   878   880   910
 init1   Rel   3518 #00DBE -  1104  1057
 pCONFG  Ext              -   850
 pERROR  Ext              -   844
 pFTYPE  Ext              -   841
 pMEM    Ext              -   847
 pMNLP   Ext              -   838
 rtnhnd  Rel   3730 #00E92 -  1311  1314
 typerr  Rel   4133 #01025 -  1469  1464
=xfrth   Rel     73 #00049 -    39
=xromF0  Rel     70 #00046 -    36
```


Input Parameters

  Source file name is MR&FT0

  Listing file name is MR/FT0::65

  Object file name is MR%FT0::65

```
                                    111111
                          0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
                TITLE   BASIC-->FORTH KEYWORDS
                RDSYMB  MR%GTO

FLOATF  EQU     5               STATUS BIT USED FOR FLOAT FLAG
REV?    EQU     2               1ST OR 2ND TIME THROUGH NEXTOK
NUMFLG  EQU     1               NUMBER OR STRING
FUN     EQU     0               FUNCTION FLAG
*********************************
*
* PUT PARAMETERS ON FORTH STACK
*
*********************************
*
* R0[S]=PARAMETER COUNT SET IN PFSET
* IF FROM FORTHF, R0[6-2]=PTR INTO TEMP BUFFER
*

00000 850       FNSET2  ST=1    FUN
00003 7C94      FNSET3  GOSUB   SAVDX       SAVE D1,D0
00007 24                P=      4
00009 8F00              GOSBVL  =R<RSTK
      000
        * SET ACTIVE FLAG; CLEAR ERR? FLAG
00010 1BBB              D0=(5)  =oERR?
      BF2
00017 D2                C=0     A
00019 144               DAT0=C  A               CLEAR ERR?
0001C 1BC0              D0=(5)  =ACTIVE
      BF2
00023 E6                C=C+1   A
00025 144               DAT0=C  A               SET ACTIVE

        * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        * FROM HERE TO TLOOP IS JUST TO SET SIZE OF FORTH  STACK
        *
00028 7F84              GOSUB   GETDX       RESTORE D1, D0
0002C 34DF              LC(5)   (=DSTKAD)   GET CURRENT STACK BASE
      AF2
00033 137               CD1EX
00036 DA                A=C     A           SAVE D1
00038 147               C=DAT1  A           GET FORTH TOS
0003B 133               AD1EX               RESTORE D1
0003E 134               D0=C                USE D0 TO SET FORTH TOS
00041 137               CD1EX
00044 D7                D=C     A           SAVE D1 AGAIN
00046 135               D1=C
00049 118               C=R0                GET PARAMETER COUNTER
0004C AE2               C=0     B           #PARMS IN C[S];ZERO OUR COUNTER
0004F 108               R0=C
00052 852               ST=1    REV?        REVERSE STRINGS THIS TIME
        *
        * LOOP TO SET STACK SIZE
        * IN THIS LOOP D,D0&D1 ARE WHAT SHOULD BE SAVED
        *
00055 7601      ST1     GOSUB   NEXTOK      LOOK AT TOKEN
```



```
00059 72E1              GOSUB   PCOUNT      DONE WITH PARAMETERS?
0005D 4B1               GOC     ST2
        * ADJUST D0 FOR FORTH TOS
00060 861               ?ST=0   NUMFLG      STRING?
00063 70                GOYES   STX1        IF SO WE WANT TO SUBTRACT 10
00065 875               ?ST=1   FLOATF      FLOATING?
00068 DE                GOYES   ST1         FLOAT NUM DOESN'T CHANGE INT STACK
0006A 184       STX1    D0=D0-  5           ELSE PUT ANOTHER ITEM ON FORTH STK
0006D 871               ?ST=1   NUMFLG      NUM?
00070 5E                GOYES   ST1
00072 184               D0=D0-  5           EXTRA 5 FOR STRING
00075 6FDF              GOTO    ST1
        ***************************************************
        * NOW ACTUALLY PUT THINGS ON STACKS IN TLOOP
        *
00079 DB        ST2     C=D     A           RESTORE BASIC TOS
0007B 135               D1=C
0007E 136               CD0EX
00081 D7                D=C     A           D=FORTH TOS
00083 10B               R3=C                SAVE FOR SETBAS
00086 AC2               C=0     S
00089 109               R1=C                ZERO F.P. COUNTER
        * reset temp param counter R0[B]
0008C 118               C=R0
0008F AE2               C=0     B
00092 108               R0=C
00095 842               ST=0    REV?            STRINGS ALREADY REVERSED
        *
        * IN TLOOP WE CARE ABOUT A&B (DATA ITEMS) AND D (STK PTR)
        * AS WELL AS D1 WHICH WE WILL STORE AWAY IN FNSET1
        * R0[S,B] ARE COUNTERS;R0[2-6]=IOBUF PTR
        * R1 IS STR PTR WHEN NEXTOK HITS STRING;R1[S]=F.P.COUNTER
        * R2 IS SCRATCH USED BY PCOUNT
        * R3[A] IS BEING SAVED FOR SETBAS
        *
00098 73C0      TLOOP   GOSUB   NEXTOK
0009C 7F91              GOSUB   PCOUNT      COUNT PARAMETERS
000A0 412               GOC     FPSET?      DONE DOING PARAMETERS
000A3 861               ?ST=0   NUMFLG      STRING?
000A6 E0                GOYES   PUTSTR
        * NUMERIC PARAMETER
000A8 7890              GOSUB   PUTSTK      PUT IT ON INT OR F.P. STACK
000AC 6BEF              GOTO    TLOOP
000B0 6C71      DATER   GOTO    DATERR

000B4 7190      PUTSTR  GOSUB   PUTINT      PUT A ON STK
000B8 D4                A=B     A
000BA 7B80              GOSUB   PUTINT
000BE 69DF              GOTO    TLOOP
        * ------------------------------------------------
000C2 865       FPSET?  ?ST=0   FLOATF
000C5 54                GOYES   FNSET1
        * NOW PUT F.P. ITEMS IN X,Y,Z,T FROM TEMP BUFFER
000C7 137               CD1EX
000CA 10A               R2=C                    SAVE D1 FROM STKLFT
```

```
107 000CD 111          A=R1                    GET COUNTER IN S
108 000D0 118          C=R0
109 000D3 BF6          CSR     W
110 000D6 BF6          CSR     W               GET BUFFER PTR IN C[A]
111 000D9 134          D0=C
112           * PTR WAS LEFT AT LAST ENTRY + 16
113 000DC 18F          D0=D0-  16
114 000DF 948  FPSTO   ?A=0    S               NO MORE PARAMS?
115 000E2 E2           GOYES   FNST1+          R2 ALREADY HAS D1
116 000E4 8F00         GOSBVL  =STKLFT         USES A,C,D1
          000
117 000EB 1F0D         D1=(5)  =oX
          BF2
118 000F2 1567         C=DAT0  W
119 000F6 18F          D0=D0-  16              POINT TO NEXT ENTRY
120 000F9 1557         DAT1=C  W
121 000FD 111          A=R1
122 00100 A4C          A=A-1   S               DECREMENT COUNTER
123 00103 101          R1=A
124 00106 68DF         GOTO    FPSTO
125           ************************************************
126           *
127           * PARAMETERS ARE ON STACK -- NOW GET COMMAND STRING
128           *
129 0010A 137  FNSET1  CD1EX
130 0010D 10A          R2=C                    SAVE D1
131 00110 77A3 FNST1+  GOSUB   GETDX
132 00114 11A          C=R2                    RESTORE D1
133 00117 135          D1=C
134 0011A 7583         GOSUB   SAVDX
135           * SAVED ORIGINAL D0,AND D1 AFTER PARAMS POPPED
136 0011E 119          C=R1                    GET PTR TO STR FROM NEXTOK
137 00121 135          D1=C                    SET FOR POP1S
138 00124 8F00         GOSBVL  =POP1S          A=LEN;D1-->STR
          000
139 0012B 137          CD1EX
140 0012E D7           D=C     A               D=STR PTR FOR SETBAS
141 00130 81C          ASRB                    NIBS-->CHARS (A SAFE FROM RSTK<R)
142 00133 24           P=      4
143 00135 8F00         GOSBVL  =RSTK<R         RESTORE RTN STACK
          000
144 0013C D8           B=A     A               B=LEN FOR SETBAS
145 0013E 7202         GOSUB   SETBAS
146 00142 01           RTN
147
148 00144 875  PUTSTK  ?ST=1   FLOATF          FLOATING?
149 00147 41           GOYES   putflt
150
151 00149 DB   PUTINT  C=D     A               GET SP
152 0014B 134          D0=C
153 0014E 140          DAT0=A  A               PUT A ON STK
154 00151 164          D0=D0+  5
155 00154 136          CD0EX                   SAVE TOS
156 00157 D7           D=C     A               D=SP
157 00159 01           RTN
```



```
158
159 0015B 6D80 putflt  GOTO    PUTFLT
160
161
162           ****************************************
163           *
164           * NEXTOK GETS ITEM FROM STACK
165           * ENTRY: D1 POINTS TO STACK
166           * EXIT: D1 POINTS TO NEXT ITEM, P=0
167           *       S1=1 FOR NUM, 0 FOR STRING
168           *       R1 POINTS TO STR IF IT WAS ONE
169           * USES: A,B,C,R1
170           ****************************************
171 0015F 1537 NEXTOK  A=DAT1  W
172 00163 851          ST=1    NUMFLG          FLAG NUM
173 00166 309          LC(1)   9
174 00169 986          ?A>C    P               NOT REAL?
175 0016C F1           GOYES   NT01
176 0016E 875          ?ST=1   FLOATF          FLOATING?
177 00171 B0           GOYES   ENDNT           THEN JUST LEAVE AS IS
178 00173 8F00         GOSBVL  =FLTDH          ELSE UNFLOAT IT IN A
          000
179 0017A 04           SETHEX
180 0017C 17F  ENDNT   D1=D1+  16
181 0017F 137          CD1EX
182 00182 135          D1=C                    RESTORE D1
183 00185 01           RTN
184 00187 65A0 daterr  GOTO    DATERR
185 0018B 31E0 NT01    LCHEX   0E              COMPLEX?
186 0018F 962          ?A=C    B
187 00192 5F           GOYES   daterr
188 00194 B04          A=A+1   P
189 00197 A64          A=A+A   B
190 0019A 96C          ?A#0    B
191 0019D AE           GOYES   daterr          DOPEVECTOR
192           * STRING
193 0019F 862          ?ST=0   REV?            STR WAS REVERSED 1ST TIME
194 001A2 71           GOYES   STRNT           THROUGH HERE
195           *  FIRST REVERSE IT (IT'S BACKWARDS)
196 001A4 DB           C=D     A               SAVE D
197 001A6 06           RSTK=C                  ON RTNSTK
198 001A8 8F00         GOSBVL  =REV$           REVERSE STRING
          000
199 001AF 1537         A=DAT1  W               GET BACK HEADER
200 001B3 20           P=      0
201 001B5 07           C=RSTK                  RETRIEVE D (SP)
202 001B7 D7           D=C     A
203           * UPDATE D1
204 001B9 841  STRNT   ST=0    NUMFLG          FLAG STRING
205 001BC AF8          B=A     W
206 001BF F5           BSR     A
207 001C1 F5           BSR     A               B[A]=LENGTH
208 001C3 119          C=R1                    SAVE COUNT IN R1[S]
209 001C6 137          CD1EX
210 001C9 109          R1=C                    SAVE STR PTR FOR FNSET1
```

```
211 001CC DA          A=C     A           GET HEADER ADDR
212 001CE 133         AD1EX
213 001D1 17F         D1=D1+  16          POINT TO STR ITSELF
214 001D4 133         AD1EX               A=STR PTR
215 001D7 C9          C=C+B   A           ADVANCE D1 BY STRING LENGTH
216 001D9 81D         BSRB                NIBS-->CHARS FOR FORTH
217 001DC 135         D1=C
218 001DF D9          C=B     A
219 001E1 D8          B=A     A
220 001E3 DA          A=C     A           SWITCH B&A SO LEN GOES ON 1ST
221 001E5 669F        GOTO    ENDNT
222
223           * PUT REAL # IN A ON FORTH F.P.STACK
224           * PUT IN TEMP BUFFER, POINTED TO BY R0[6-2]
225           * UPDATE COUNTER IN R1[S]
226 001E9 AF8 PUTFLT  B=A     W               SAVE DATA ITEM
227 001EC 111         A=R1                    GET COUNTER
228 001EF 304         LC(1)   4               ONLY COUNT UP TO 4
229 001F2 816         CSRC                    PUT IN C[S]
230 001F5 942         ?A=C    S
231 001F8 00          RTNYES                  IGNORE ALL BUT LAST 4
232 001FA B44         A=A+1   S
233 001FD 101         R1=A                    UPDATE COUNTER
234 00200 118         C=R0
235 00203 BF6         CSR     W
236 00206 BF6         CSR     W
237 00209 134         D0=C
238 0020C AF4         A=B     W
239 0020F 1507        DAT0=A  W               PUT ITEM IN BUFFER
240 00213 16F         D0=D0+  16              POINT TO NEXT SPACE
241 00216 132         ABEX
242 00219 BF0         ASL     W
243 0021C BF0         ASL     W
244 0021F 118         C=R0
245 00222 AEA         A=C     B               PRESERVE R0[S,B]
246 00225 ACA         A=C     S
247 00228 100         R0=A
248 0022B 01          RTN
249
250 0022D 8F00 DATERR GOSBVL  =BYE1
        000
251 00234 3100        LC(2)   =eDATTY
252 00238 8D00        GOVLNG  =BSERR
        000
253
254           **********************
255           * RETURN CARRY SET IF DONE WITH PARAMS
256           *
257 0023F 870 PCOUNT  ?ST=1   FUN
258 00242 72          GOYES   FUNCNT
259           * FROM FORTHX -- JUST DETERMINE IF WE'RE DONE
260           * BY CHECKING AGAINST BOTTOM OF STACK
261
262 00244 102         R2=A                    SAVE DATA ITEM
263 00247 133         AD1EX                   SAVE D1
```



```
264 0024A 1F00        D1=(5)  =TFORN          BOTTOM OF STACK
        000
265 00251 147         C=DAT1  A
266 00254 8A2         ?A=C    A               THERE YET?
267 00257 A0          GOYES   BOT
268 00259 131         D1=A
269 0025C 112         A=R2                    RESTORE DATA ITEM
270 0025F 03          RTNCC
271 00261 131 BOT     D1=A
272 00264 112         A=R2                    "
273 00267 02          RTNSC
274
275           * FOR FUNCTIONS, USE COUNTER IN R0
276 00269 102 FUNCNT  R2=A                    SAVE DATA ITEM
277 0026C 110         A=R0                    A[S]=# OF PARMS;A[B]=COUNT
278 0026F AF6         C=A     W
279 00272 B66         C=C+1   B
280 00275 108         R0=C                    UPDATE TEMP COUNTER
281 00278 816         CSRC                    PUT COUNT IN S
282 0027B 942         ?A=C    S
283 0027E 70          GOYES   BOT2
284 00280 112         A=R2
285 00283 03          RTNCC
286 00285 112 BOT2    A=R2
287 00288 02          RTNSC
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER

```
288                     EJECT
289             *
290             * FORTHX
291             *
292             * Execute string parameter as line of FORTH words.
293             * Take up to 14 extra parameters, which go to integer stack.
294
295 0028A 8D00 FRTHXd  GOVLNG  =FIXDC       (REPLACES  DELAYd)
        000
296 00291 AC2  FRTHXp  C=0     S
297 00294 10A          R2=C                 PARAM COUNTER
298 00297 8F00 NPAR    GOSBVL  =EXPPAR
        000
299 0029E 4D2          GOC     SYNTER           DUMMY ARRAY
300 002A1 870          ?ST=1   0                INVALID EXPRESSION?
301 002A4 82           GOYES   SYNTER
302 002A6 11A          C=R2                 GET COUNTER
303 002A9 94E          ?C#0    S
304 002AC 70           GOYES   NP01
305             * FIRST TIME, CHECK FOR STRING
306 002AE 873          ?ST=1   3                NOT STR?
307 002B1 B1           GOYES   SYNTER
308 002B3 B46  NP01    C=C+1   S                COUNT
309 002B6 451          GOC     SYNTER       >15
310 002B9 10A          R2=C                 UPDATE COUNTER
311             * DEMAND COMMA OR END OF STATEMENT
312 002BC 3100         LC(2)   =tCOMMA
313 002C0 962          ?A=C    B
314 002C3 4D           GOYES   NPAR
315 002C5 8D00         GOVLNG  =RESPTR
        000
316
317 002CC 8D00 SYNTER  GOVLNG  =SYNTXe
        000
318
319 002D3 7BFF         REL(5)  FRTHXd
        F
320 002D8 9BFF         REL(5)  FRTHXp
        F
321
322 002DD 8F00 =FORTHX GOSBVL  =EXPEXC          PUT VALUES ON MTHSTK
        000
323 002E4 7BB1         GOSUB   SAVDX        SAVE FROM XFTH
324 002E8 8E00         GOSUBL  =XFTH
        00
325 002EE 1BC0         D0=(5)  =ACTIVE      FORTHRAM NOW IN PLACE
        BF2
326 002F5 146          C=DAT0  A            GET VALUE OF ACTIVE
327 002F8 CE           C=C-1   A
328 002FA CE           C=C-1   A            ACTIVE=2? (BASICF/I/$)
329 002FC 8AA          ?C=0    A
330 002FF F1           GOYES   BXERR
331 00301 CE           C=C-1   A            ACTIVE=3? (BASICX)
332 00303 8AA          ?C=0    A
333 00306 81           GOYES   BXERR
```



```
334 00308 7FA1         GOSUB   GETDX
335 0030C 845          ST=0    FLOATF       FLAG NOT FLOATING PT
336 0030F 840          ST=0    FUN          FLAG STATEMENT, NOT FUNCTION
337 00312 7DEC         GOSUB   FNSET3
338 00316 7C00         GOSUB   BXRTN
339 0031A 6ED0         GOTO    FX
340
341 0031E 3104 BXERR   LC(2)   =eFORTX   "FORTH NOT RE-ENTRANT"
342 00322 6171         GOTO    ERRSET
343
344             * * *
345 00326 7301 BXRTN   GOSUB   BRTN
346             * RETURN HERE
347 0032A 8F00 FXRTN   GOSBVL  =BYE1          CLEAR ACTIVE,
        000
348             *                            RESTORE PRGM STATUS
349 00331 7681         GOSUB   GETDX
350 00335 8D00         GOVLNG  =NXTSTM
        000
351
```

```
                        EJECT
                *********************************************************
                * CODE PREPARED FOR BY FNSET2, FNSET1 (WHICH CALLS IT)...
                * SET UP COMMAND STRING IN TIB
                * ZERO BLK AND IN
                * SET PTR TO BASIC, FILLED IN AT FX
                *********************************************************
003C 3153 SWF   LC(2)   =eSWF
0340 6351       GOTO    ERRSET

0344 1F61 SETBAS D1=(5) =oR0        RTN STK PTR
     BF2
034B D4         A=B     A          GET LEN
034D C4         A=A+A   A          IN NIBS
034F 1B61       D0=(5)  =oTIB
     BF2
0356 146        C=DAT0  A          C=CURRENT TIB
0359 108        R0=C               SAVE IT
035C C2         C=C+A   A          END OF WHERE STRING WILL GO
035E 143        A=DAT1  A          GET RTN STK PTR (WE WILL RESET IT BEL
0361 8B2        ?A<C    A          WILL IT TRASH RTN STK?
0364 8D         GOYES   SWF
0366 11B        C=R3               GET STACK PTR
0369 135        D1=C
036C 1C4        D1=D1-  5
036F DB         C=D     A          get ptr to string back
0371 145        DAT1=C  A          push "from" addr for NMOVE
0374 118        C=R0               GET BACK CURRENT TIB
0377 1C4        D1=D1-  5
037A 145        DAT1=C  A          push "to" addr (=TIB)
037D D9         C=B     A          get length
037F 108        R0=C               save length (R0,R3 safe from NMOVE)
0382 C6         C=C+C   A          make nibs for NMOVE
0384 1C4        D1=D1-  5
0387 145        DAT1=C  A          push length
038A 8F00       GOSBVL  =$NMOVE
     000
                * put in two nulls
0391 20         P=      0
0393 1FB1       D1=(5)  (=oTIB)
     BF2
039A 147        C=DAT1  A          CURRENT TIB
039D 110        A=R0               get length of string
03A0 C4         A=A+A   A          chars-->nibs
03A2 C2         C=C+A   A
03A4 135        D1=C
03A7 D2         C=0     A
03A9 145        DAT1=C  A
                * clear the LCD in case something is to be shown
03AC 26         P=      6
03AE 8F00       GOSBVL  =R<RSTK
     000
                *       GOSBVL  =CRLFOF
                * ZERO BLOCK AND IN
03B5 20         P=      0
```



```
03B7 D0         A=0     A
03B9 3425       LC(5)   (=oBLK)
     BF2
03C0 135        D1=C
03C3 141        DAT1=A  A
03C6 3475       LC(5)   (=oIN)
     BF2
03CD 135        D1=C
03D0 141        DAT1=A  A
03D3 26         P=      6
03D5 8F00       GOSBVL  =RSTK<R
     000
03DC 1F61       D1=(5)  =oR0
     BF2
03E3 147        C=DAT1  A
03E6 D5         B=C     A          SET RTN STK PTR
03E8 113        A=R3               GET TOS IN A
03EB 20         P=      0
03ED 342A       LC(5)   (=oBASIC)
     BF2
03F4 135        D1=C
03F7 01         RTN
```

```
419                     EJECT
420 003F9 145  FX       DAT1=C  A           PUT RTN IN oBASIC
421 003FC 133           AD1EX               RESTORE D1
422
423 003FF 8E00          GOSUBL  =SETI
          00
424 00405 0000          CON(5)  =INTRP
          0
425 0040A 0000          CON(5)  =BASIC
          0
426 0040F 0000          CON(5)  =AT          PUT RTN ADDR ON STACK
          0
427 00414 0000          CON(5)  =ZERO
          0
428 00419 0000          CON(5)  =BASIC
          0
429 0041E 0000          CON(5)  =STORE      ZERO OUT BASIC
          0
430 00423 0000          CON(5)  =?STK
          0
431 00428 0000          CON(5)  =BSCRTN
          0
432             *
433             * THIS ROUTINE IS IN THE HARD ROM
434             *=BSCRTN CON(5)  =$BSR
435             *=$BSR   C=DAT1  A
436             *        D1=D1+  5
437             *        D0=(5)  =DSTKAD
438             *        CD1EX
439             *        DAT0=C  A
440             *        CD1EX
441             *        RSTK=C
442             *        RTN
443             *
444             *
445             * RETURN FROM FORTH ENVIRONMENT
446             *
447 0042D 07   BRTN    C=RSTK
448 0042F 01           RTN
449
```



```
450                     EJECT
451             ******************************************
452             *
453             * FORTHI
454             * Same as FORTHX but returns an integer.
455             *
456             ******************************************
457
458 00431 00              NIBHEX  00          NO PARAMETERS
459 00433 7C60 =FORTHI    GOSUB   SAVDX
460 00437 8E00            GOSUBL  =FTHRM?       DOES FORTH EXIST?
          00
461 0043D 5A4             GONC    NOFRTH        NO
462 00440 8E00            GOSUBL  =GOOD?      IN RIGHT PLACE?
          00
463 00446 514             GONC    NOFRTH
464 00449 1B11            D0=(5)  =oS0
          BF2
465 00450 146             C=DAT0  A             BOTTOM OF STACK
466 00453 1BDF            D0=(5)  =DSTKAD       GET TOS
          AF2
467 0045A 142             A=DAT0  A
468 0045D 8BE             ?A>=C   A
469 00460 03              GOYES   EMPSTK        EMPTY STACK
470 00462 131             D1=A
471 00465 143             A=DAT1  A             GET ITEM
472 00468 174             D1=D1+  5
473 0046B 137             CD1EX
474 0046E 144             DAT0=C  A             RESET TOS
475 00471 8F00            GOSBVL  =HDFLT        FLOAT NUMBER
          000
476 00478 04              SETHEX
477 0047A 7D30            GOSUB   GETDX
478 0047E AF6             C=A     W           RESULT IN C
479 00481 8D00            GOVLNG  =FNRTN1     check mem, rtn answer
          000
480
481
482 00488 3121 NOFRTH     LC(2)   =eNOFTH
483 0048C 6700            GOTO    ERRSET
484 00490 31E0 EMPSTK     LC(2)   =eEMPT
485 00494 22   ERRSET     P=      2
486 00496 31F2            LC(2)   =ID
487 0049A 20              P=      0
488 0049C 8D00 GOERR      GOVLNG  =BSERR
          000
489
490
491             * SAVE D0,D1 AT FUNCR0
492 004A3 137  SAVDX      CD1EX
493 004A6 1F00            D1=(5)  =FUNCR0
          000
494 004AD 145             DAT1=C  A
495 004B0 174             D1=D1+  5
496 004B3 136             CD0EX
```

```
497 004B6 145          DAT1=C  A
498 004B9 01           RTN
499             * RESTORE D1,D0
500 004BB 1B00 GETDX   D0=(5)  =FUNCR0
          000
501 004C2 146          C=DAT0  A
502 004C5 137          CD1EX
503 004C8 164          D0=D0+  5
504 004CB 146          C=DAT0  A
505 004CE 136          CD0EX
506 004D1 01           RTN
507
```



```
508                    EJECT
509             *
510             * FORTHF
511             *
512             * Same as FORTHI but uses floating point stack
513             *
514 004D3 00            NIBHEX  00
515 004D5 7ACF =FORTHF GOSUB   SAVDX
516 004D9 8E00         GOSUBL  =FTHRM?
          00
517 004DF 58A          GONC    NOFRTH
518 004E2 8E00         GOSUBL  =GOOD?
          00
519 004E8 5F9          GONC    NOFRTH
520 004EB 1B0D         D0=(5)  =oX
          BF2
521 004F2 1527         A=DAT0  W           GET RESULT
522 004F6 71CF         GOSUB   GETDX
523 004FA AF6          C=A     W
524 004FD 8D00         GOVLNG  =FNRTN1     check mem, rtn answer
          000
525
```

```
                        EJECT
                *
                * FORTH$
                *
                * Return string to BASIC
                *
00504 638F GNFRTH   GOTO    NOFRTH
00508 00           NIBHEX  00
0050A 759F =FORTH$ GOSUB   SAVDX
0050E 8E00         GOSUBL  =FTHRM?
      00
00514 5FE          GONC    GNFRTH
00517 8E00         GOSUBL  =GOOD?
      00
0051D 56E          GONC    GNFRTH
00520 1B11         D0=(5)  =oS0        BOS
      BF2
00527 146          C=DAT0  A
0052A 136          CD0EX
0052D 189          D0=D0-  10
00530 136          CD0EX               C=STK WITH 2 THINGS
00533 1BDF         D0=(5)  =DSTKAD
      AF2
0053A 142          A=DAT0  A           GET TOS
0053D 8BA          ?A<=C   A
00540 60           GOYES   F$1         OK
00542 6D4F         GOTO    EMPSTK
00546 131  F$1     D1=A
00549 AF2          C=0     W           CLEAR FOR CSRB BELOW
0054C 147          C=DAT1  A               GET TOP ITEM (LEN)
0054F 174          D1=D1+  5
00552 143          A=DAT1  A               GET NEXT (ADDR)
00555 174          D1=D1+  5
00558 103          R3=A                R3=ADDR
0055B 133          AD1EX               A=NEW TOS
0055E 140          DAT0=A  A           UPDATE FORTH TOS
00561 113          A=R3
00564 C6           C=C+C   A           NIBS FOR STRHDR
00566 10A          R2=C                SAVE LEN
00569 7E4F         GOSUB   GETDX
0056D 11A          C=R2
00570 8F00         GOSBVL  =STRHDR
      000
00577 11A          C=R2                GET BACK CHAR COUNT*2
0057A 132          AD0EX
0057D 102          R2=A                SAVE D0
00580 113          A=R3                GET STR PTR
00583 CA           A=A+C   A           POINT TO END OF STR
00585 130          D0=A
00588 81E          CSRB                NIBS-->CHARS AGAIN
0058B 8AA          ?C=0    A
0058E 51           GOYES   FIN$
                * STR IS BACKWARDS
00590 181  PUTST   D0=D0-  2           POINT TO NEXT CHAR (LAST FIRST)
00593 14A          A=DAT0  B           GET A CHAR
```



```
00596 149          DAT1=A  B           PUT IT ON STACK
00599 171          D1=D1+  2           POINT TO NEXT CHAR
0059C CE           C=C-1   A           COUNT DOWN CHARS
0059E 8AE          ?C#0    A
005A1 FE           GOYES   PUTST
005A3 111  FIN$    A=R1                GET TOS LEFT BY STRHDR
005A6 131          D1=A
005A9 112          A=R2                GET BACK D0
005AC 130          D0=A
005AF 8D00         GOVLNG  =EXPR
      000

005B6 00   =FILEND NIBHEX 00

005B8              END
```

$NMOVE	Ext		-	385							
?STK	Ext		-	430							
AT	Ext		-	426							
BASIC	Ext		-	425	428						
BOT	Rel	609	#00261 -	271	267						
BOT2	Rel	645	#00285 -	286	283						
BRTN	Rel	1069	#0042D -	447	345						
BSCRTN	Ext		-	431							
BSERR	Ext		-	252	488						
BXERR	Rel	798	#0031E -	341	330	333					
BXRTN	Rel	806	#00326 -	345	338						
BYE1	Ext		-	250	347						
DATER	Rel	176	#000B0 -	95							
DATERR	Rel	557	#0022D -	250	95	184					
EMPSTK	Rel	1168	#00490 -	484	469	548					
ENDNT	Rel	380	#0017C -	180	177	221					
ERRSET	Rel	1172	#00494 -	485	342	360	483				
EXPEXC	Ext		-	322							
EXPPAR	Ext		-	298							
EXPR	Ext		-	585							
F$1	Rel	1350	#00546 -	549	547						
=FILEND	Rel	1462	#005B6 -	587							
FIN$	Rel	1443	#005A3 -	581	572						
FIXDC	Ext		-	295							
FLOATF	Abs	5	#00005 -	4	57	102	148	176	335		
FLTDH	Ext		-	178							
FNRTN1	Ext		-	479	524						
FNSET1	Rel	266	#0010A -	129	103						
FNSET2	Rel	0	#00000 -	18							
FNSET3	Rel	3	#00003 -	19	337						
FNST1+	Rel	272	#00110 -	131	115						
=FORTH$	Rel	1290	#0050A -	534							
=FORTHF	Rel	1237	#004D5 -	515							
=FORTHI	Rel	1075	#00433 -	459							
=FORTHX	Rel	733	#002DD -	322							
FPSET?	Rel	194	#000C2 -	102	89						
FPSTO	Rel	223	#000DF -	114	124						
FRTHXd	Rel	650	#0028A -	295	319						
FRTHXp	Rel	657	#00291 -	296	320						
FTHRM?	Ext		-	460	516	535					
FUN	Abs	0	#00000 -	7	18	257	336				
FUNCNT	Rel	617	#00269 -	276	258						
FUNCRO	Ext		-	493	500						
FX	Rel	1017	#003F9 -	420	339						
FXRTN	Rel	810	#0032A -	347							
GETDX	Rel	1211	#004BB -	500	33	131	334	349	477	522	561
GNFRTH	Rel	1284	#00504 -	532	536	538					
GOERR	Rel	1180	#0049C -	488							
GOOD?	Ext		-	462	518	537					
HDFLT	Ext		-	475							
INTRP	Ext		-	424							
NEXTOK	Rel	351	#0015F -	171	51	87					
NOFRTH	Rel	1160	#00488 -	482	461	463	517	519	532		
NP01	Rel	691	#002B3 -	308	304						
NPAR	Rel	663	#00297 -	298	314						

NT01	Rel	395	#0018B -	185	175						
NUMFLG	Abs	1	#00001 -	6	55	60	90	172	204		
NXTSTM	Ext		-	350							
PCOUNT	Rel	575	#0023F -	257	52	88					
POP1S	Ext		-	138							
PUTFLT	Rel	489	#001E9 -	226	159						
PUTINT	Rel	329	#00149 -	151	97	99					
PUTST	Rel	1424	#00590 -	574	580						
PUTSTK	Rel	324	#00144 -	148	93						
PUTSTR	Rel	180	#000B4 -	97	91						
R<RSTK	Ext		-	21	398						
RESPTR	Ext		-	315							
REV$	Ext		-	198							
REV?	Abs	2	#00002 -	5	46	78	193				
RSTK<R	Ext		-	143	410						
SAVDX	Rel	1187	#004A3 -	492	19	134	323	459	515	534	
SETBAS	Rel	836	#00344 -	362	145						
SETI	Ext		-	423							
ST1	Rel	85	#00055 -	51	58	61	63				
ST2	Rel	121	#00079 -	67	53						
STKLFT	Ext		-	116							
STORE	Ext		-	429							
STRHDR	Ext		-	563							
STRNT	Rel	441	#001B9 -	204	194						
STX1	Rel	106	#0006A -	59	56						
SWF	Rel	828	#0033C -	359	371						
SYNTER	Rel	716	#002CC -	317	299	301	307	309			
SYNTXe	Ext		-	317							
TFORN	Ext		-	264							
TLOOP	Rel	152	#00098 -	87	94	100					
XFTH	Ext		-	324							
ZERO	Ext		-	427							
daterr	Rel	391	#00187 -	184	187	191					
eDATTY	Ext		-	251							
putflt	Rel	347	#0015B -	159	149						
tCOMMA	Ext		-	312							

Input Parameters

Source file name is MR&FTB

Listing file name is MR/FTB::65

Object file name is MR%FTB::65

```
                                  111111
                        0123456789012345
```

Initial flag settings are

Errors

None

Saturn Assembler News

```
 1                      TITLE  Jump_Table,_0,1,&2_char_words
 2                      RDSYMB MR%GIO
 3 E0000                ABS    #E0000
 4
 5                ************************************************
 6                *                                              *
 7                * MR&FR1:    This is the first assembly in the *
 8                *         hard addressed FORTH.  It contains the *
 9                *         ROM word jump table and 0, 1, and 2 char *
10                *         words.                               *
11                *                                              *
12                ************************************************
13
14                *      ***  FIND'S SPEED TABLE  ***
15
16 E0000 6400 =TABLE  CON(5) =!X             THE NULL WORD
         E
17 E0005 4050         CON(5) =!QUOTE         1 CHAR WORDS
         E
18 E000A 8F11         CON(5) =!S<            2 CHAR WORDS
         E
19 E000F 0000         CON(5) =!ASC           3 CHAR WORDS
         0
20 E0014 0000         CON(5) =!END$          4 CHAR WORDS
         0
21 E0019 0000         CON(5) =!LEFT$         5 CHAR WORDS
         0
22 E001E 0000         CON(5) =!RITE$         6 CHAR WORDS
         0
23 E0023 0000         CON(5) =!FADJ          7 CHAR WORDS
         0
24 E0028 0000         CON(5) =!CLALL         8 CHAR WORDS
         0
25 E002D 0000         CON(5) =!IMMED         9 CHAR WORDS
         0
26 E0032 0000         CON(5) =!VOCAB         10 CHAR WORDS
         0
27 E0037 0000         CON(5) =!DEFIN         11 CHAR WORDS
         0
28 E003C 0000         CON(5) =!STRAR         12 CHAR WORDS
         0
```



```
29                      EJECT
30                ************************************************
31                *
32                * X null word
33                *
34                * FIND "recognizes" this word of length zero
35                * when WORD has hit its terminator.
36                * INTERPRET then calls EXECUTE which branches
37                * to this routine. Its main purpose is to pop
38                * the return to the repeat-branch in INTERPRET,
39                * landing us up a level in QUIT.
40                * When we are interpreting from a screen (file),
41                * we need to check that we aren't in an
42                * unfinished colon definition, because WORD doesn't
43                * quit until the EOF when it's in a file (whereas
44                * you get here at the end of every line from the
45                * terminal/keyboard).
46                *
47                ************************************************
48
49 E0041 0000           CON(5)  0              No more 0 length words!
         0
50 E0046 0C    =!X     CON(2)  #C0            IMMEDIATE
51 E0048 08            CON(2)  #80
52 E004A 0000 =X       CON(5) =DOCOL
         0
53 E004F 0000         CON(5) =BLK            are we working from a screen?
         0
54 E0054 1720         CON(5) =AT
         E
55 E0059 0000         CON(5) =ZBRNH          branch if not
         0
56 E005E 6100         CON(5) (X1)-*
         0
57 E0063 0000         CON(5) =STATE          we're working from a screen and it
         0
58 E0068 1720         CON(5) =AT             EOF;  if we're still in compile
         E
59 E006D 0000         CON(5) =ABRT"X         mode then we have an error conditi
         0
60 E0072 30           CON(2)  =eNO;          we'll give "FTH ERR: no ending ;"
61                *    NIBASC  \no endin\     message
62                *    NIBASC  \g  ;\
63 E0074 0F90 X1      CON(5) =R>             pop one level from return stack
         E
64 E0079 0000         CON(5) =DROP           throw it away
         0
65 E007E 0000         CON(5) =SEMI           return
         0
```

```
                    EJECT
                ***********************************************
                *                                             *
                *          START OF 1 CHAR WORDS*             *
                *       ADD NEW DEFINITIONS AT THIS END       *
                *                                             *
                ***********************************************

                ******************************************
                *
                *  X  ( --- addr )  This rtne returns the
                *       addr of the X register in the floating
                *       point stack.
                *
                ******************************************
E0083 0000              CON(5) 0
      0
E0088 18        =!XX    CON(2) #81
E008A 8D                CON(2) \X\+#80
E008C 1900      =XX     CON(5) =$XX
      E

E0091 8E15      =$XX    GOSUBL =DOUSE
      C0
E0097 0DBF              CON(5) (=oX)
      2


                ******************************************
                *
                *  Y  ( --- addr ) This rtne returns the
                *       addr of the Y register in the floating
                *       point stack.
                *
                ******************************************
E009C 8800              CON(5) =!XX
      E
E00A1 18        =!Y     CON(2) #81
E00A3 9D                CON(2) \Y\+#80
E00A5 AA00      =Y      CON(5) =$Y
      E

E00AA 8E83      =$Y     GOSUBL =DOUSE
      C0
E00B0 0EBF              CON(5) (=oY)
      2


                ******************************************
                *
                * Z  ( --- addr)  This rtne returns the
                *       addr of the Z register in the floating
                *       point stack.
                *
                ******************************************
```



```
E00B5 1A00              CON(5) =!Y
      E
E00BA 18        =!Z     CON(2) #81
E00BC AD                CON(2) \Z\+#80
E00BE 3C00      =Z      CON(5) =$Z
      E

E00C3 8EF1      =$Z     GOSUBL =DOUSE
      C0
E00C9 0FBF              CON(5) (=oZ)
      2


                ******************************************
                *
                * T ( --- addr) This rtne returns the addr
                *       of the T register in the floating
                *       point stack.
                *
                ******************************************
E00CE AB00              CON(5) =!Z
      E
E00D3 18        =!T     CON(2) #81
E00D5 4D                CON(2) \T\+#80
E00D7 CD00      =T      CON(5) =$T
      E

E00DC 8E60      =$T     GOSUBL =DOUSE
      C0
E00E2 0CCF              CON(5) (=oT)
      2
```

```
                        EJECT
                *********************************************
                *
                * L ( --- addr) This rtne returns the addr
                *       of the L register in the floating
                *       point stack.
                *
                *********************************************
E00E7 3D00              CON(5) =!T
      E
E00EC 18        =!L     CON(2) #81
E00EE CC                CON(2) \L\+#80
E00F0 5F00      =L      CON(5) =$L
      E

E00F5 8EDE      =$L     GOSUBL =DOUSE
      B0
E00FB 0CBF              CON(5) (=oLASTX)
      2
```



```
                        EJECT
                *********************************************************************
                *
                * ' (TIC)  ( --- addr)  Leave compilation address of next word
                *
                *********************************************************************

E0100 CE00              CON(5)  =!L
      E
E0105 18        =!TIC   CON(2)  #81              '
E0107 7A                CON(2)  \'\+#80
E0109 0000      =TIC    CON(5)  =DOCOL           :
      0
E010E 0000              CON(5)   =-FIND          search dictionary for word
      0

                * -FIND returns cfa ch flag

E0113 58A0              CON(5)   =ZEQ            reverse sense of flag
      E
E0118 0000              CON(5)   =ABORTW         ABORT" not found"
      0
E011D 20                CON(2)  =eNOTF
E011F 0000              CON(5)   =DROP           drop the character,
      0
                *                                leaving just the cfa
E0124 0000              CON(5)   =LITR           if compiling, literal will
      0
                *                                compile in code which will
                *                                return cfa to stack at
                *                                runtime
E0129 0000              CON(5)   =SEMI           ;
      0
                *********************************************
                *
                * I  ( --- n)  Copy loop index to data stack
                *
                * In a loop, the index is supposed to be the
                * top value on the return stack. So this word
                * is the same as R@, except it only works in
                * a definition.
                *
                * IMMEDIATE, COMPILE ONLY
                *
                *********************************************

E012E 5010              CON(5)  =!TIC
      E
E0133 1C        =!I     CON(2)  #C1              IMMEDIATE
E0135 9C                CON(2)  \I\+#80
E0137 0000      =II     CON(5)  =DOCOL           R@ DOES SAME FUNCTION
      0
E013C 0000              CON(5)  =COMPL           enforce compile mode
      0
E0141 04A0              CON(5)  =R@              copy top value of rtn stk to data
```

```
                E
195 E0146 0000          CON(5)  =SEMI           ;
          0
```



```
196                     EJECT
197             ************************************************************
198             *
199             * J  ( --- n)  returns the value of the 2nd loop index
200             *    contained on the RTN STK.  This routine
201             *    only makes sense in a colon definition in the middle
202             *    of 2 DO LOOPs.
203             *
204             *    IMMEDIATE, COMPILE ONLY
205             *
206             *                            LIMIT 2
207             *                            INDEX 2
208             *                            LIMIT 1
209             *      RTN STK PTR --------> INDEX 1
210             *
211             ************************************************************
212
213 E014B 3310            CON(5)  =!I
              E
214 E0150 1C    =!J       CON(2)  #C1             IMMEDIATE
215 E0152 AC              CON(2)  \J\+#80
216 E0154 0000  =J        CON(5)  =DOCOL          :
              0
217 E0159 0000            CON(5)  =COMPL          enforce compile mode
              0
218 E015E 8610            CON(5)  =JJ             J RUN TIME ROUTINE
              E
219 E0163 0000            CON(5)  =SEMI           ;
              0
220
221 E0168 D610  =JJ       CON(5)  $JJ
              E
222 E016D D9    $JJ       C=B     A                 copy rtn stk to C(A)
223 E016F 137             CD1EX                     D1=RTN STK, C=DATA STK
224 E0172 179             D1=D1+  10                POINT TO 2ND INDEX
225 E0175 143             A=DAT1  A                 A=2ND INDEX
226 E0178 135             D1=C                      RECOVER DATA STK TO D1
227 E017B 1C4             D1=D1-  5
228 E017E 141             DAT1=A  A                 PUSH INDEX TO DATA STACK
229 E0181 03              RTNCC
```

```
                        EJECT
                *****************************************************
                *
                * ADD  (n1 n2 --- n3 ) leave sum (n1+n2) on stack
                *
                *****************************************************

E0183 0510              CON(5)  =!J
      E
E0188 18    =!ADD       CON(2)  #81
E018A BA                CON(2)  \+\+#80
E018C 1910  =ADD        CON(5)  =$ADD
      E

E0191 147   =$ADD       C=DAT1  A               POP n2
E0194 174               D1=D1+  5
E0197 143               A=DAT1  A               POP n1
E019A C2                C=C+A   A               add 'em!
E019C 145               DAT1=C  A               PUSH sum
E019F 03                RTNCC
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER


```
                        EJECT
                *****************************************
                *
                * MINUS  (n1 n2 --- n3)
                *
                * Leave difference (n1-n2) on stack.
                *
                *****************************************

E01A1 8810              CON(5)  =!ADD
      E
E01A6 18    =!MINUS     CON(2)  #81             -
E01A8 DA                CON(2)  \-\+#80

E01AA FA10  =MINUS      CON(5)  =$MINUS
      E
E01AF 147   =$MINUS     C=DAT1  A               POP n2
E01B2 174               D1=D1+  5
E01B5 143               A=DAT1  A               POP n1
E01B8 EA                A=A-C   A               n1-n2
E01BA 141               DAT1=A  A           RETURN DIFFERENCE TO STACK
E01BD 03                RTNCC
```

```
                        EJECT
                *********************************************
                *
                * =  (n1 n2 --- flag) true if n1=n2 else flase
                *
                *********************************************

E01BF 6A10          CON(5)  =!MINUS
      E
E01C4 18    =!EQUAL CON(2)  #81             =
E01C6 DB            CON(2)  \=\+#80
E01C8 DC10  =EQUAL  CON(5)  =$EQUAL
      E

E01CD 143   =$EQUAL A=DAT1  A               POP n2
E01D0 174           D1=D1+  5
E01D3 147           C=DAT1  A               POP n1
E01D6 E2            C=C-A   A
E01D8 D0            A=0     A               PREPARE FOR FALSE
E01DA 8AE           ?C#0    A               EQUAL?
E01DD 40            GOYES   EQ0             NO
E01DF CC            A=A-1   A               TRUE, (ALL 1'S)
E01E1 141   EQ0     DAT1=A  A               WRITE FLAG
E01E4 03            RTNCC
```



```
                        EJECT
                *********************************************
                *
                * LT  (n1 n2 --- flag)  True if n1 is less than n2
                *
                *********************************************

E01E6 4C10          CON(5)  =!EQUAL
      E
E01EB 18    =!LT    CON(2)  #81
E01ED CB            CON(2)  \<\+#80
E01EF 4F10  =LT     CON(5)  =$LT
      E

E01F4 143   =$LT    A=DAT1  A               POP n2
E01F7 174           D1=D1+  5
E01FA 147           C=DAT1  A               POP n1
E01FD 25    GTLT1   P=      5
E01FF A80           A=0     P               CLEAR HIGH NIB FOR SIGN SHIFT
E0202 A82           C=0     P
E0205 A14           A=A+A   WP              n2*2
E0208 A16           C=C+C   WP              n1*2
E020B 902           ?C=A    P               DO n1 & n2 HAVE SAME SIGN?
E020E 50            GOYES   SAMSGN          YES, THEN COMPARE NOW
E0210 A9E           ACEX    WP              DIFFERENT SIGNS, SWITCH SENSE
                *                           OF TEST
E0213 996   SAMSGN ?A>C     WP              n2 > n1 ?
E0216 90            GOYES   TRUE            YES
E0218 D0    FALSE   A=0     A               RETURN FALSE (0) FLAG
E021A 141           DAT1=A  A               STORE FLAG
E021D 03            RTNCC

E021F D0    TRUE    A=0     A
E0221 CC            A=A-1   A               TRUE = ALL 1'S
E0223 141           DAT1=A  A               STORE FLAG ON STACK
E0226 03            RTNCC
```

```
                        EJECT
                *****************************************************
                *
                * GT (n1 n2 --- flag)  True if n1 greater than n2
                *
                *****************************************************

E0228 BE10          CON(5)  =!LT
      E
E022D 18    =!GT    CON(2)  #81
E022F EB            CON(2)  \>\+#80
E0231 6320  =GT     CON(5)  =$GT
      E

E0236 147   =$GT    C=DAT1  A               POP n2
E0239 174           D1=D1+  5
E023C 143           A=DAT1  A               POP n1
E023F 6DBF          GOTO    GTLT1           FINISH IN LT
```



```
                        EJECT
                *****************************************
                *
                * !  (n addr --- )   store n at addr.
                *
                *****************************************

E0243 D220          CON(5)  !GT
      E
E0248 18    =!STORE CON(2)  #81             !
E024A 1A            CON(2)  \!\+#80
E024C 1520  =STORE  CON(5)  =$STORE
      E

E0251 147   =$STORE C=DAT1  A               pop addr
E0254 174           D1=D1+  5
E0257 143           A=DAT1  A               read n into A
E025A 174           D1=D1+  5               return nothing on stack
E025D 137           CD1EX                   ADDR TO D1, SAVE DATA STK IN C
E0260 141           DAT1=A  A               store n in addr
E0263 135           D1=C                    DATA STK PTR to D1
E0266 03            RTNCC                   all done!
```

```
                        EJECT
                *******************************************************
                *
                * @  (addr --  n)   replace addr with contents of addr
                *
                *******************************************************

E0268 8420          CON(5)  =!STORE
      E
E026D 18    =!AT    CON(2)  #81
E026F 0C            CON(2)  \@\+#80
E0271 6720  =AT     CON(5)  =$AT
      E

E0276 147   =$AT    C=DAT1  A               pop addr
E0279 137           CD1EX                   addr to D1
E027C 143           A=DAT1  A               contents of addr
E027F 135           D1=C                    D1=DATA STK PTR,
E0282 141           DAT1=A  A               PUSH CONTENTS TO STACK
E0285 03            RTNCC
```



```
                        EJECT

                ***************************************
                *
                * 0 ( -- 0) push  a 0 to the DATA STK
                *
                ***************************************

E0287 D620          CON(5)  !AT
      E
E028C 18    =!ZERO  CON(2)  #81              0
E028E 0B            CON(2)  \0\+#80

E0290 0000  =ZERO   CON(5)  =DOCON
      0
E0295 0000          CON(5)  0
      0

                ***************************************
                *
                * 1 ( -- 1)  Put a 1 on the DATA STK
                *
                ***************************************

E029A C820          CON(5)  !ZERO
      E
E029F 18    =!ONE   CON(2)  #81
E02A1 1B            CON(2)  \1\+#80

E02A3 0000  =ONE    CON(5)  =DOCON
      0
E02A8 1000          CON(5)  1
      0
```

```
                    EJECT

              ****************************************
              *
              * 2 ( -- 2)  Put a 2 on the DATA STK
              *
              ****************************************

E02AD F920          CON(5)  =!ONE
      E
E02B2 18    =!TWO   CON(2)  #81
E02B4 2B            CON(2)  \2\+#80

E02B6 0000  =TWO    CON(5)  =DOCON
      0
E02BB 2000          CON(5)  2
      0


              ************************************************
              *
              * THREE   ( --- 3) leaves a 3 on the data stack
              *
              ************************************************

E02C0 2B20          CON(5)  =!TWO
      E
E02C5 18    =!THREE CON(2)  #81              3
E02C7 3B            CON(2)  \3\+#80

E02C9 0000  =THREE  CON(5)  =DOCON
      0
E02CE 3000          CON(5)  3
      0
```



```
                    EJECT
              ********************************************************
              *
              * COMMA  ( n --- )  Allot 5 NIBS in the dictionary,
              *        storing n there.
              *
              * -----VECTORED WORD-----
              *
              ********************************************************

E02D3 5C20          CON(5)  =!THREE
      E
E02D8 18    =!COMMA CON(2)  #81              COMMA ,
E02DA CA            CON(2)  \,\+#80
E02DC 0000  =COMMA  CON(5)  =DOCOL           :
      0
E02E1 0000          CON(5)  =LIT
      0
E02E6 F2CF          CON(5)  =oCOMMA          VECTOR FOR COMMA
      2
E02EB 1720          CON(5)  =AT              CFA TO USE
      E
E02F0 0000          CON(5)  =EXEC            EXECUTE
      0
E02F5 0000          CON(5)  =SEMI            ;
      0

              *
              * DEFAULT COMMA ROUTINE
              *

E02FA 0000  =ICOMMA CON(5)  =DOCOL           :
      0
E02FF 0000          CON(5)   =HERE           first free addr in dictionary
      0
E0304 C420          CON(5)   =STORE          store n HERE
      E
E0309 0000          CON(5)   =LIT
      0
E030E 5000          CON(5)   5                 5
      0
E0313 0000          CON(5)   =ALLOT          update DP by 5
      0
E0318 0000          CON(5)   =SEMI           ;
      0
```

```
                        EJECT
                *******************************************************
                *
                * /  (DIVIDE)  (n1 n2 --- n3)  Divide n1 by n2 leave
                *       quotient n3.
                *
                *******************************************************

E031D 8D20E             CON(5)  =!COMMA
E0322 18        =!/     CON(2)  #81
E0324 FA                CON(2)  \/\+#80
E0326 00000     =/      CON(5)  =DOCOL          :

E032B 00000             CON(5)   =/MOD           /MOD
                * /MOD leaves remainder quotient
E0330 00000             CON(5)   =SWAP      quotient remainder
E0335 00000             CON(5)   =DROP      drop the remainder
E033A 00000             CON(5)   =SEMI      leave quotient

                *************************************************************
                *
                * [  (left bracket) Text from input stream is subsequently
                *    executed.
                *
                *  IMMEDIATE
                *
                *************************************************************

E033F 2230E             CON(5)  =!/
E0344 1C        =!LBRAC CON(2)  #C1             [
E0346 BD                CON(2)  \[\+#80

E0348 00000 =LBRAC  CON(5)  =DOCOL          :
E034D 0920E             CON(5)   =ZERO          set value of user variable
E0352 00000             CON(5)   =STATE         STATE to 0
E0357 C420E             CON(5)   =STORE
E035C 00000             CON(5)   =SEMI           ;
```



```
                        EJECT

                *******************************************************
                *
                * .  (DOT)  (n --- ) Display the top number on the stack
                *
                *******************************************************

E0361 4430E             CON(5)  =!LBRAC
E0366 18        =!DOT   CON(2)  #81
E0368 EA                CON(2)  \.\+#80

E036A 00000 =DOT    CON(5)  =DOCOL          : .
E036F 00000             CON(5)   =S->D      MAKE DOUBLE NUMBER
E0374 DBE0E             CON(5)   =D.        DISPLAY IT
E0379 00000             CON(5)   =SEMI          ;
```

```
                EJECT
*****************************************************************
*
* # (SHARP)     (ud1 --- ud2)  Generate from an unsigned double
*               number (ud1) the next ASCII char which is placed
*               in an output string.  ud2 is the quotient after
*               division by BASE and is maintained for further
*               processing.  Used between <# and #>.
*
*****************************************************************

E037E 6630          CON(5)  !DOT
      E
E0383 18    =!N#    CON(2)  #81             #
E0385 3A            CON(2)  \#\+#80
E0387 0000  =N#     CON(5)  =DOCOL          : #
      0
E038C 0000          CON(5)   =BASE          ud1
      0
E0391 1720          CON(5)   =AT            ud1 b
      E
E0396 0000          CON(5)   =M/MOD         u2 ud2
      0
E039B 0000          CON(5)   =ROT           ud2 u2
      0
E03A0 0000          CON(5)   =LIT
      0
E03A5 9000          CON(5)  9               ud2 u 9
      0
E03AA 0000          CON(5)   =OVER          ud2 u 9 u
      0
E03AF FE10          CON(5)   =LT            ud2 u (9<u)
      E
E03B4 0000          CON(5)   =ZBRNH         IF greater than 9 add 7
      0
E03B9 4100          CON(5)  (N#00)-*
      0

            * ASCII "0"=48; A=10 (in higher base) needs to
            * be offset by 7+48=55 because ASCII "A"=10+55=65

E03BE 0000          CON(5)   =LIT
      0
E03C3 7000          CON(5)  7               7
      0
E03C8 C810          CON(5)   =ADD           +
      E

E03CD 0000  N#00    CON(5)   =LIT
      0

            * ASCII "0"=48

E03D2 0300          CON(5)  #30             48
```



```
      0
E03D7 C810          CON(5)   =ADD           +
      E
E03DC 0000          CON(5)   =HOLD      put in numeral temp space
      0
E03E1 0000          CON(5)   =SEMI          ;
      0
```

```
559                     EJECT
560                 **********************************
561                 *
562                 * * (n1 n2 --- n3)  (MULTIPLY)
563                 *
564                 **********************************
565
566 E03E6 3830         CON(5)  =!N#
        E
567 E03EB 18    =!MULT  CON(2)  #81             *
568 E03ED AA            CON(2)  \*\+#80
569
570 E03EF 0000 =MULT   CON(5)  =DOCOL          :
        0
571 E03F4 0000         CON(5)   =U*        leaves double number
        0
572 E03F9 0000         CON(5)   =DROP      drop the high 5 nibbles
        0
573 E03FE 0000         CON(5)   =SEMI           ;
        0
```



```
574                     EJECT
575
576                 ******************************************************
577                 *
578                 * ; (SEMICOLON) Terminate a colon definition and stop
579                 *               compilation.
580                 *
581                 ******************************************************
582 E0403 BE30         CON(5)  =!MULT
        E
583 E0408 1C    =!SEMIC CON(2)  #C1             ;  (IMMEDIATE)
584 E040A BB            CON(2)  \;\+#80
585
586 E040C 0000 =SEMIC  CON(5)  =DOCOL          :
        0
587 E0411 0000         CON(5)   =QCSP          ?CSP make sure that stack is
        0
588                 *                           at the same level as when we
589                 *                           started compilation of this
590                 *                           definition.
591
592 E0416 0000         CON(5)   =COMPL         COMPILE  runtime exit routine
        0
593 E041B 0000         CON(5)   =SEMI          ;  (REPLACES EXIT)
        0
594
595 E0420 0000         CON(5)   =SMUDG         unsmudges (SMUDGE is a toggle)
        0
596                 *                           this definition so it can be foun
597                 *                           in word searches
598
599 E0425 8430         CON(5)   =LBRAC         set STATE to zero, this restores
        E
600                 *                           execute mode
601 E042A 0000         CON(5)   =SEMI          ;
        0
```

```
                        EJECT
                *****************************************************
                *
                * ] (RIGHT BRACKET)  Sets compilation mode so that
                *   subsequent text from the input stream is compiled.
                *
                *****************************************************
E042F 8040E     =!SEMIC CON(5)  !SEMIC
E0434 18        =!RBRAC CON(2)  #81             ]
E0436 DD                CON(2)  \]\+#80
E0438 00000     =RBRAC  CON(5)  =DOCOL          :
E043D 00000             CON(5)   =LIT
E0442 0C000             CON(5)  #C0             #C0
E0447 00000             CON(5)   =STATE         STATE
E044C C420E             CON(5)   =STORE         !
E0451 00000             CON(5)   =SEMI          ;
                *
                * THE POINT OF STORING #C0 IS FOR A COMPARE
                * IN INTERPRET WITH THE FIRST BYTE OF A WORD
                * HEADER -- IF IT'S >#C0 THEN IT IS IMMEDIATE.
                * SO IMMEDIATE WORDS EXECUTE ANYTIME, THE FIRST
                * BYTE OF OTHER NAME FIELDS ARE < #C0 AND THEY
                * ARE COMPILED RATHER THAN EXECUTED.
                *
```



```
                        EJECT
                ***************************************
                *
                *  :    ( - )    used as     : <name>
                * This is "colon" for colon-definitions!
                * First remember top-of-stack so ";" can check
                * that it is the same at the end of the definition.
                * Then make sure the defining-context is the
                * search-context so this word gets entered as
                * the proper "LATEST" word in the CURRENT vocabulary.
                *
                *********************************************

E0456 4340E     =!COLON CON(5)  =!RBRAC
E045B 18        =!COLON CON(2)  #81
E045D AB                CON(2)  \:\+#80
E045F 00000     =COLON  CON(5) =DOCOL
E0464 00000             CON(5)  =SP@            store away current top of stack
E0469 00000             CON(5)  =CSP            into the user variable CSP
E046E C420E             CON(5)  =STORE
E0473 00000             CON(5)  =CURR           make the current vocabulary the
E0478 1720E             CON(5)  =AT             context (or search ) vocabulary
E047D 00000             CON(5)  =CONTX
E0482 C420E             CON(5)  =STORE
E0487 00000             CON(5)  =CREAT          CREATE a dictionary entry for <name
E048C 00000             CON(5)  =SMUDG          SMUDGE its name field so it can't
                *                               be recursively defined.
E0491 8340E             CON(5)  =RBRAC          ] set compilation mode
E0496 00000             CON(5)  =ROM;C          ROM;CODE  insert prologue
E049B 00000             CON(5) =DOCOL           docolon
```

```
                    EJECT

            ****************************************
            *
            * ?  (addr --- ) display value at addr.
            *
            ****************************************

E04A0 B540          CON(5)  =!COLON
      E
E04A5 18    =!QUEST CON(2)  #81          ?
E04A7 FB            CON(2)  \?\+#80

E04A9 0000  =QUEST  CON(5)  =DOCOL       :
      0
E04AE 1720          CON(5)   =AT         @
      E
E04B3 A630          CON(5)   =DOT        .
      E
E04B8 0000          CON(5)   =SEMI       ;
      0
```



```
                    EJECT
            ****************************************************************
            *
            * (  PAREN      Set comment mode in either compile or execution m
            *               insists on closing ")" or will give error message
            *
            ****************************************************************

E04BD 5A40          CON(5)  =!QUEST
      E
E04C2 1C    =!PAREN CON(2)  #C1          (
E04C4 8A            CON(2)  \(\+#80

E04C6 0000  =PAREN  CON(5)  =DOCOL       :
      0
E04CB 0000          CON(5)   =LIT
      0
E04D0 EFFF          CON(5)  (0-2)        -2
      F
E04D5 0000          CON(5)   =IN         >IN
      0
E04DA 6AA0          CON(5)   =PSTOR      +!
      E
            * MOVED POINTER BACK ONE CHAR FOR GCHAR, IN CASE
            * THERE WAS JUST "( )"
E04DF 0000          CON(5)   =LIT
      0
E04E4 9200          CON(5)  #29          ")"
      0
E04E9 0000          CON(5)   =GCHAR      moves string to HERE
      0
            *                            tries to match )
            *                            returns 1 if no match
            *                            also string addr.
E04EE 0000          CON(5)   =ABRT"X
      0
E04F3 40            CON(2)  =eNOEP
            *       NIBASC   \no endin\
            *       NIBASC   \g  )\
E04F5 0000          CON(5)   =DROP        don't need string addr
      0
E04FA 0000          CON(5)   =SEMI       ;
      0
```

```
                    EJECT

              ****************************************
              * "
              * QUOTE                      ( -- str )
              *
              * Get string constant from input stream,
              * create a temporary string on the PAD (execute mode),
              * or compile run-time action and string into dictionary.
              *
              * string= {maxlen} {len} {chars}
              *
              * String on PAD has a MAXLEN of 80 or length of
              * string, whichever is greater; string compiled into
              * dictionary must of course have maxlen=len=just what
              * the actual length is.
              *
              ****************************************

E04FF 2C40          CON(5)  =!PAREN
      E
E0504 1C    =!QUOTE CON(2)  #C1          IMMEDIATE
E0506 2A            CON(2)  \"\+#80

E0508 0000  =QUOTE  CON(5)  =DOCOL
      0
E050D 0000          CON(5)  =STATE       if contents of STATE
      0
E0512 1720          CON(5)  =AT          are non-zero then we
      E
              *                          are compiling.

E0517 0000          CON(5)  =ZBRNH       IF we are compiling
      0
E051C 9100          CON(5)  (QUOT2)-*
      0
E0521 0000          CON(5)  =COMPL       then COMPILE the runtime
      0
E0526 0460          CON(5)  =QUOTC       quote word QUOTE-C into dictionary;
      E
E052B 6B20          CON(5)  =TWO         ALLOT 2 nibs for the string's
      E
E0530 0000          CON(5)  =ALLOT       maxlen byte
      0

E0535 0000  QUOT2   CON(5)  =NQOT?       NULL STRING?  (NQOT? returns
      0
              *                          a true if next char is quote, returns
              *                          false otherwise.  i.e. " " is a null
              *                          string.

E053A 0000          CON(5)  =ZBRNH        IF null then
      0
E053F 8200          CON(5)  (QUOT02)-*    we simply
      0
```



```
E0544 0000          CON(5)  =PT1+        point past the ending "
      0
E0549 0920          CON(5)  =ZERO        put a 0 on data stack
      E
E054E 0000          CON(5)  =HERE        HERE points to current-len byte
      0
E0553 CCC0          CON(5)  =C!          set current-length byte to 0
      E
E0558 0000          CON(5)  =HERE       LEAVE HERE ON STACK AS GCHAR DOES
      0
E055D 0000          CON(5)  =BRNCH
      0
E0562 B100          CON(5)  (QUOT5)-*    ELSE get string delimited by "
      0
E0567 0000  QUOT02  CON(5)  =LIT
      0
E056C 2200          CON(5)  34           "
      0
E0571 0000          CON(5)  =GCHAR       move string to HERE, look for
      0
              *                          ending ", return -1 if not there
              *                          also return string addr gained from
              *                          WORD, part of GCHAR

E0576 0000          CON(5)  =ABRT"X       if flag returned from GCHAR is not
      0
E057B 50            CON(2)  =eNO"         0 then give error message and ABORT
              *     NIBASC  \no endin\
              *     NIBASC  \g  "\

E057D 0000  QUOT5   CON(5)  =STATE       check contents of STATE
      0
E0582 1720          CON(5)  =AT          if non-zero we are compiling
      E
E0587 0000          CON(5)  =ZBRNH       IF COMPILING
      0
E058C 8200          CON(5)  (QUOT3)-*
      0
E0591 0000          CON(5)  =DUP         duplicate address of counted string
      0
E0596 9AC0          CON(5)  =CAT         get length of string
      E
E059B 0000          CON(5)  =OVER        COPY HERE (HERE points to curr len byt
      0
E05A0 49B0          CON(5)  =TWO-        MOVE BACK TO MAXLEN POSITION
      E
E05A5 CCC0          CON(5)  =C!          STORE LEN AS MAXLEN
      E
E05AA 0000          CON(5)  =MOVDP       update dictionary pointer to point
      0
              *                          beyond string.

E05AF 0000          CON(5)  =SEMI        ; CHEAT AND END HERE !
      0
```

```
783                *
784                * we are executing, not compiling; enter here with ADDR on stack;
785                * ADDR means address of string prefaced by its length in bytes
786                * (i.e., a counted string)
787                *
788 E05B4 0000 QUOT3  CON(5)  =DUP         (ADDR  ADDR)
           0
789 E05B9 E5B0        CON(5)  =TWO+        (ADDR ADDR+2)
           E
790 E05BE 0000        CON(5)  =OVER        (ADDR ADDR+2 ADDR)
           0
791                *
792                * C-COUNT means char count of string
793                *
794 E05C3 9AC0        CON(5)  =CAT         C@  (ADDR ADDR+2 C-COUNT)
           E
795 E05C8 0000        CON(5)  =PAD             (ADDR ADDR+2 C-COUNT PAD)
           0
796                *
797                * PAD+4 is our destination address
798                * ADDR+2 is our source addr
799                *
800 E05CD 0D90        CON(5)  =FOUR+       4+  (ADDR ADDR+2 C-COUNT DEST)
           E
801 E05D2 0000        CON(5)  =SWAP            (ADDR ADDR+2 DEST C-COUNT)
           0
802                *
803                * NOTE IT IS IMPORTANT THAT WE FIRST COPY THE STRING TO
804                * THE PAD BEFORE WE SET THE STRING LENGTHS AND THAT WE USE
805                * CMOVE> SINCE OTHERWISE, IF THE STRING IS LONGER THAN 39
806                * 39 CHARS WE OVERWRITE PART OF IT DURING THE MOVE.
807                *
808 E05D7 0000        CON(5)  =CMOV>       MOVE STRING TO PAD+4  (ADDR)
           0
809 E05DC 9AC0        CON(5)  =CAT         C@  (C-COUNT)
           E
810 E05E1 0000        CON(5)  =DUP         C@  (C-COUNT C-COUNT)
           0
811
812                *
813                * when we move a string to the PAD we will always set
814                * its max length to be the maximum of either 80 or its
815                * current length--this makes string operations like
816                * concatenate possible on temporary strings created with
817                * quote.
818                *
819 E05E6 0000        CON(5)  =LIT
           0
820 E05EB 0500        CON(5)  80               (C-COUNT C-COUNT 80)
           0
821 E05F0 0000        CON(5)  =MAX             (C-COUNT MAX(C-COUNT,80) )
           0
822 E05F5 0000        CON(5)  =SWAP            (MAX(CC,80) C-COUNT)
           0
823 E05FA 0000        CON(5)  =PAD             (MAX(CC,80) C-COUNT PAD)
```



```
           0
824 E05FF 0000        CON(5)  =SWAP            (MAX PAD C-COUNT)
           0
825 E0604 0000        CON(5)  =OVER            (MAX PAD C-COUNT PAD)
           0
826 E0609 E5B0        CON(5)  =TWO+        2+  (MAX PAD C-COUNT PAD+2)
           E
827 E060E CCC0        CON(5)  =C!          STORE CURR LEN  (MAX PAD)
           E
828 E0613 CCC0        CON(5)  =C!          STORE MAX LEN
           E
829                *
830                * now we will return to user the current char count of
831                * the string (contained at PAD+2) and the address of the
832                * string (contained at PAD+4).
833                *
834 E0618 0000        CON(5)  =PAD
           0
835 E061D E5B0        CON(5)  =TWO+            ( PAD+2)
           E
836 E0622 0000        CON(5)  =DUP             ( PAD+2 PAD+2 )
           0
837 E0627 9AC0        CON(5)  =CAT             ( PAD+2 C-COUNT )
           E
838 E062C 0000        CON(5)  =SWAP
           0
839 E0631 E5B0        CON(5)  =TWO+             COUNT PAD+4
           E
840 E0636 0000        CON(5)  =SWAP
           0
841 E063B 0000        CON(5)  =SEMI        ;   all done, go home
           0
842
843                *
844                * RUNTIME ACTION COMPILED INTO WORD
845                *
846
847 E0640 0000 =QUOTC CON(5)  =DOCOL       :
           0
848 E0645 04A0        CON(5)  =R@          R@  return address
           E
849                *                       has already been incremented to
850                *                       point at the max len byte of string.
851
852 E064A 0000        CON(5)  =DUP         adr adr
           0
853 E064F 0D90        CON(5)  =FOUR+       adr adr+4  this gives address of actua
           E
854 E0654 0000        CON(5)  =SWAP        adr+4 adr
           0
855 E0659 9AC0        CON(5)  =CAT         adr+4 count   get current string lengt
           E
856 E065E 0000        CON(5)  =DUP         adr+4 count count
           0
857 E0663 E5B0        CON(5)  =TWO+        adr+4 count count+2  char count plus
```

```
          E
858             *                       2 length bytes
859 E0668 6DA0          CON(5)  =TWO*   adr+4 count 2*(count+2)
          E
860             *                       this gives us the nibble count of stri
861             *                       and length bytes
862 E066D 0F90          CON(5)  =R>     R>  pop addr of maxlen byte from RTN S
          E
863 E0672 C810          CON(5)  =ADD    +  point beyond string
          E
864 E0677 81A0          CON(5)  =>R     >R  push this addr as next addr to int
          E
865             *
866             * LEAVE ADR+4 COUNT on data stack...i.e.,the string
867             *
868 E067C 0000          CON(5)  =SEMI   done, go home!
          0
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER


```
869                     STITLE  2 CHAR WORDS
870             ********************************************
871             *                                          *
872             *          START OF 2 CHAR WORDS           *
873             *      ADD ADDITIONAL WORDS AT THIS END    *
874             *                                          *
875             ********************************************
876             *
877             ******************************
878             *
879             * FP    FRACTIONAL PART
880             *       FLOATING POINT WORD,
881             *       OPERATES ON X REGISTER
882             *
883             ******************************
884
885 E0681 0000            CON(5)  0
          0
886 E0686 28    =!FP      CON(2)  #82
887 E0688 64              NIBASC  \F\
888 E068A 0D              CON(2)  \P\+#80
889 E068C 1960  =FP       CON(5)  =$FP
          E
890 E0691 8E00  =$FP      GOSUBL  =NUMST          SAVEFP;X IN (A,B)
          00
891
892             * Get the value stored in FORTRAM at (=oX) into
893             * the CPU registers expected by mainframe routine.
894
895 E0697 822             SB=0
896 E069A 821             XM=0
897 E069D 8F00            GOSBVL  =FRAC15
          000
898 E06A4 6442            GOTO    =FEND           PUTABX;GETFP
899
900             ******************************************
901             *
902             * IP            FLOATING POINT WORD
903             *
904             * INTEGER PART -- OPERATE ON X REGISTER
905             *
906             ******************************************
907
908 E06A8 6860            CON(5)  =!FP
          E
909 E06AD 28    =!IP      CON(2)  #82
910 E06AF 94              NIBASC  \I\
911 E06B1 0D              CON(2)  \P\+#80
912 E06B3 8B60  =IP       CON(5)  =$IP
          E
913 E06B8 8E00  =$IP      GOSUBL  =NUMST
          00
914 E06BE 822             SB=0
915 E06C1 821             XM=0
916 E06C4 8F00            GOSBVL  =CLRFRC
```

```
            000
917 E06CB 6D12          GOTO    =FEND
```



```
                        EJECT
                *********************************************
                *
                * .S   This word implements a non-destructive
                *      stack print.  NOTE: it does an unsigned
                *      stack display.
                *
                *********************************************

E06CF DA60          CON(5)  =!IP
      E
E06D4 28    =!DOTS  CON(2)  #82
E06D6 E2            CON(2)  \.\
E06D8 3D            CON(2)  \S\+#80
E06DA 0000  =DOTS   CON(5)  =DOCOL          :
      0
E06DF C3F0          CON(5)  =CR             ISSUE CARRIAGE RETURN
      E
E06E4 0000          CON(5)  =DEPTH          FIND # ITEMS
      0
E06E9 58A0          CON(5)  =ZEQ            IF EMPTY STACK
      E
E06EE 0000          CON(5)  =ABRT"X
      0
E06F3 E0            CON(2)  =eEMPT          ABORT" empty stack"
E06F5 0000          CON(5)  =SP@            STACK LEVEL NOW
      0
E06FA E3E0          CON(5)  =S0             STACK START
      E
E06FF 0000          CON(5)  =SWAP           SWITCH FOR LOOP
      0
E0704 0000          CON(5)  =XDO            PREPARE LOOP
      0
E0709 04A0  DS00    CON(5)  =I              GET ADDR
      E
E070E 1720          CON(5)  =AT             CONTENTS OF ADDR
      E
E0713 0920          CON(5)  =ZERO           MAKE IT DOUBLE WORD
      E
E0718 DBE0          CON(5)  =D.             SHOW IT
      E
E071D 0000          CON(5)  =LIT            INCREMENT BY 1 ADDR's WORTH
      0
E0722 5000          CON(5)  5
      0
E0727 0000          CON(5)  =<+LP>          +LOOP
      0
E072C DDFF          CON(5)  (DS00)-*
      F
E0731 0000          CON(5)  =SEMI           ;
      0
```

```
                        EJECT
                *****************************************
                *
                * H.  (n --- ) Display n in base 16
                *      as an unsigned number.
                *
                *****************************************
E0736 4D60E     =!H.    CON(5)  =!DOTS
E073B 28        =!H.    CON(2)  #82
E073D 84                CON(2)  \H\
E073F EA                CON(2)  \.\+#80
E0741 00000     =H.     CON(5)  =DOCOL
E0746 00000             CON(5)  =BASE           GET CURRENT BASE
E074B 1720E             CON(5)  =AT
E0750 00000             CON(5)  =SWAP           n TO TOP OF STACK
E0755 00000             CON(5)  =HEX            SET HEX MODE
E075A EA01E             CON(5)  =U.             DISP AS UNSIGNED NUMBER
E075F 00000             CON(5)  =BASE           RESTORE BASE
E0764 C420E             CON(5)  =STORE
E0769 00000             CON(5)  =SEMI           ; DONE
```



```
                        EJECT
                *****************************************
                *
                * LN  This word takes the natural log of
                *     the value in the X register of the
                *     floating point stack and puts the answer
                *     into X and puts X into LASTX.
                *
                *****************************************
E076E B370E             CON(5)  !H.
E0773 28        =!LN    CON(2)  #82
E0775 C4                CON(2)  \L\
E0777 EC                CON(2)  \N\+#80
E0779 E770E     =LN     CON(5)  $LN
E077E 8E00 00   $LN     GOSUBL  =NUMST
E0784 8F00 000          GOSBVL  =LN15           TAKE NATURAL LOG
E078B 8CC5 10           GOLONG  =FEND           CLEAN UP
                *****************************************
                *
                * F.   ( --- )  This word TYPEs the contents
                *       of the floating point X register.  It
                *       unlike . it does not do a destructive
                *       type.
                *
                *****************************************
E0791 3770E             CON(5) !LN
E0796 28        =!F.    CON(2) #82
E0798 64                CON(2) \F\
E079A EA                CON(2) \.\+#80
E079C 00000     =F.     CON(5) =DOCOL
E07A1 0B70E             CON(5) =FF.             SHOW X REG
E07A6 00000             CON(5) =SPACE           ADD A SPACE
E07AB 00000             CON(5) =SEMI            DONE

                *
                * REAL F.
                *

E07B0 5B70E     =FF.    CON(5) $F.
E07B5 7B00 00   =$F.    GOSUB  $XSTR$           see below
E07B9 8C00 00           GOLONG =$TYPE           show user result
```

```
1013            * This routine is used by F. and FSTR$.  It makes
1014            * a string from the contents of the X register.
1015
1016
1017 E07BF 4C70 =XSTR$   CON(5) $XSTR$
         E
1018 E07C4 8E00 $XSTR$   GOSUBL =SAVEFP          SAVE FORTH POINTERS
         00
1019 E07CA 8F00          GOSBVL =COLLAP          COLLAPSE MATH STACK
         000
1020 E07D1 1F00          D1=(5) =MTHSTK
         000
1021 E07D8 143           A=DAT1 A                A(A)=TOP OF STACK
1022 E07DB 131           D1=A                    D1-->T.O.S.
1023 E07DE 20            P=     0
1024 E07E0 3401          LC(5)  16
         000
1025 E07E7 EA            A=A-C  A
1026 E07E9 1B00          D0=(5) =AVMEMS          MAKE SURE THERE'S
         000
1027 E07F0 146           C=DAT0 A                ROOM FOR 16 NIBS
1028 E07F3 8B6           ?A>C   A                ROOM?
1029 E07F6 90            GOYES  F.0              YES
1030 E07F8 8D00          GOVLNG =MEMERR          NO, GIVE ERROR
         000
1031
1032 E07FF 1CF  F.0      D1=D1- 16
1033 E0802 1B0D          D0=(5) =oX
         BF2
1034 E0809 1567          C=DAT0 W                C(W)=X
1035 E080D 1557          DAT1=C W                WRITE X TO MATH STACK
1036 E0811 8F00          GOSBVL =STR$SB          CONVERT TO STRING
         000
1037 E0818 8F00          GOSBVL =REV$            REVERSE STRING
         000
1038 E081F 171           D1=D1+ 2
1039 E0822 AF2           C=0    W
1040 E0825 147           C=DAT1 A                READ LENGTH IN NIBS
1041 E0828 81E           CSRB                    LENGTH IN BYTES
1042 E082B 17D           D1=D1+ 14               D1-->STR
1043 E082E 06            RSTK=C                  SAVE BYTE COUNT
1044 E0830 133           AD1EX
1045 E0833 8E00          GOSUBL =GETFP           RESTORE FORTH POINTERS
         00
1046 E0839 1C4           D1=D1- 5
1047 E083C 141           DAT1=A A                WRITE STR ADDR TO DATA STK
1048 E083F 1C4           D1=D1- 5
1049 E0842 07            C=RSTK
1050 E0844 145           DAT1=C A                WRITE BYTE CNT TO DATASTK
1051 E0847 01            RTN                     returns addr-cnt on stack
```



```
1052                    EJECT
1053            ********************************************
1054            *
1055            *  N@  ( addr --- n ) return the nibble quantity
1056            *       located at the given addr and put it on
1057            *       the stack.
1058            *
1059            ********************************************
1060 E0849 6970          CON(5) !F.
         E
1061 E084E 28    =!N@    CON(2) #82
1062 E0850 E4            NIBASC \N\
1063 E0852 0C            CON(2) \@\+#80
1064 E0854 9580 =N@      CON(5) $N@
         E
1065
1066 E0859 147   =$N@    C=DAT1 A                READ ADDR
1067 E085C 136           CD0EX                   ADDR TO D0
1068 E085F D0            A=0    A
1069 E0861 15A0          A=DAT0 1                READ NIBBLE
1070 E0865 134           D0=C                    RESTORE I
1071 E0868 141           DAT1=A A                WRITE NIBBLE
1072 E086B 03            RTNCC
1073
1074
1075            ********************************************
1076            *
1077            * N!  (n addr --- ) Store the nibble quantity n
1078            *     into the given addr.
1079            *
1080            ********************************************
1081 E086D E480          CON(5) =!N@
         E
1082 E0872 28    =!N!    CON(2) #82
1083 E0874 E4            NIBASC \N\
1084 E0876 1A            CON(2) \!\+#80
1085 E0878 D780 =N!      CON(5) $N!
         E
1086
1087 E087D 147   =$N!    C=DAT1 A                READ ADDR
1088 E0880 174           D1=D1+ 5
1089 E0883 15B0          A=DAT1 1                READ NIBBLE
1090 E0887 174           D1=D1+ 5                RTN NOTHING ON DATA STK
1091 E088A 136           CD0EX
1092 E088D 1580          DAT0=A 1                WRITE NIBBLE
1093 E0891 134           D0=C                    RESTORE I TO D0
1094 E0894 03            RTNCC
```

```
1095                    EJECT
1096              ********************************************
1097              *
1098              * F+            FLOATING POINT ADD
1099              *
1100              ********************************************
1101 E0896 2780          CON(5)  !N!
          E
1102 E089B 28    =!F+    CON(2)  #82
1103 E089D 64            NIBASC  \F\
1104 E089F BA            CON(2)  \+\+#80
1105 E08A1 6A80  =F+     CON(5)  =$F+
          E
1106 E08A6 7620  =$F+    GOSUB   FSTRT       SAVEFP;STKDRP;GETXLN;uMODES
1107 E08AA 8F00  NOWADD  GOSBVL  =AD15M      ADD 2 NUMBERS
          000
1108 E08B1 6730          GOTO    FEND        PUTABX;GETFP;RTNCC
1109
1110              ********************************************
1111              *
1112              * F-            FLOATING POINT SUBTRACT
1113              *
1114              ********************************************
1115 E08B5 B980          CON(5)  =!F+
          E
1116 E08BA 28    =!F-    CON(2)  #82
1117 E08BC 64            NIBASC  \F\
1118 E08BE DA            CON(2)  \-\+#80
1119 E08C0 5C80  =F-     CON(5)  =$F-
          E
1120 E08C5 7700  =$F-    GOSUB   FSTRT
1121 E08C9 BCE           C=-C-1  S               DECIMAL MODE SET BY GETXLN
1122 E08CC 6DDF          GOTO    NOWADD          IN FSTRT
1123
1124              ************************************
1125              * SUBROUTINES OF F+,F-,F*,F/       *
1126              ************************************
1127 E08D0 8E00  =FSTRT  GOSUBL  =SAVEFP     save FORTH pointers
          00
1128 E08D6 8F00          GOSBVL  =uMODES     set modes such as DEGREE/RADIAN,
          000
1129              *                           rounding,etc.
1130 E08DD 8E00          GOSUBL  =STKDRP     "drop" stack of simulated 41-C
          00
1131              *                           i.e. put Z in Y, etc.
1132 E08E3 8C00          GOLONG  =GETXLN     get contents of X in A&B, and
          00
1133              *                           put X in LASTX.
1134
1135 E08E9 8C00  =FEND   GOLONG  =PUTABX
          00
1136              * Put result of mainframe routine, left in CPU
1137              * registers A & B, back into location oX (the
1138              * "X-register"); restore FORTH pointers and return.
1139
```



```
1140              ********************************************
1141              *                                          *
1142              * F*            FLOATING POINT MULTIPLY *
1143              *                                          *
1144              ********************************************
1145 E08EF AB80          CON(5)  =!F-
          E
1146 E08F4 28    =!F*    CON(2)  #82
1147 E08F6 64            NIBASC  \F\
1148 E08F8 AA            CON(2)  \*\+#80
1149 E08FA FF80  =F*     CON(5)  =$F*
          E
1150 E08FF 7DCF  =$F*    GOSUB   FSTRT
1151 E0903 8F00          GOSBVL  =MP2-15
          000
1152 E090A 6EDF          GOTO    FEND
1153
1154              ********************************************
1155              *
1156              * F/            FLOATING POINT DIVIDE
1157              *
1158              ********************************************
1159 E090E 4F80          CON(5)  =!F*
          E
1160 E0913 28    =!F/    CON(2)  #82
1161 E0915 64            NIBASC  \F\
1162 E0917 FA            CON(2)  \/\+#80
1163 E0919 E190  =F/     CON(5)  =$F/
          E
1164 E091E 7EAF  =$F/    GOSUB   FSTRT
1165 E0922 8F00          GOSBVL  =DV2-15
          000
1166 E0929 6FBF          GOTO    FEND
1167
```

```
1168                    EJECT
1169                ************************
1170                * OF  --FROM CASE STMT   *
1171                ************************
1172
1173 E092D 3190             CON(5)  =!F/
          E
1174 E0932 2C     =!OF      CON(2)  #C2
1175 E0934 F4               NIBASC  \O\
1176 E0936 6C               CON(2)  \F\+#80
1177
1178                * Syntax check occurs to the extent that there must
1179                * be an 8 on the stack, as left at compile-time by CASE,
1180                * or a 7 as left by ENDOF. If there is an 8, it is assumed
1181                * that there are actually a pair of 8's and if 7, that the
1182                * 7 has an address with it: it is part of the general
1183                * scheme of control structures (including LEAVE) that things
1184                * on the stack at compile-time come in pairs.
1185                *
1186                * Note that there is no way to ensure that the value to
1187                * be matched at runtime has actually been compiled at this
1188                * point.
1189
1190 E0938 0000 =OF    CON(5)  =DOCOL      :
          0
1191 E093D 0000            CON(5)  =?COMP      ensure that we are in compile mode
          0
1192 E0942 0000            CON(5)   =DUP
          0
1193 E0947 0000            CON(5)   =LIT
          0
1194 E094C 8000            CON(5)   8          TO MATCH CASE'S 8
          0
1195 E0951 8C10             CON(5)  =EQUAL
          E
1196 E0956 0000             CON(5)  =ZBRNH         IF
          0
1197 E095B 4100             CON(5)  (OF01)-*
          0
1198 E0960 0000             CON(5)  =2DROP         DROP 2 8'S LEFT BY CASE
          0
1199 E0965 0000             CON(5)  =BRNCH        ELSE
          0
1200 E096A 0200             CON(5)  (OF02)-*
          0
1201                * MUST BE A 7 FROM ENDOF OR ABORT
1202 E096F 0000 OF01        CON(5)  =DUP           DUP POSSIBLE 7
          0
1203 E0974 0000             CON(5)  =LIT
          0
1204 E0979 7000             CON(5)  7
          0
1205 E097E 5801             CON(5)  =<>            not equal
          E
1206 E0983 0000             CON(5)  =ABORTW
```



```
          0
1207 E0988 11               CON(2)  =eCOND      "conditionals not paired"
1208                * LEFT 7 AND ADDR FROM ENDOF IF WE SAW 7
1209                * (don't disturb them)
1210
1211 E098A 0000 OF02        CON(5)   =COMPL     COMPILE
          0
1212                *                                    (at runtime:)
1213 E098F 0000             CON(5)   =OVER      OVER  get the index value
          0
1214 E0994 0000             CON(5)   =COMPL     COMPILE
          0
1215 E0999 8C10             CON(5)   =EQUAL     =     does it match value preceding t
          E
1216 E099E 0000             CON(5)   =COMPL     COMPILE
          0
1217 E09A3 0000             CON(5)   =ZBRNH     0BRANCH  if not jump to ENDOF or ENDC
          0
1218                *                                       (addr to be filled in later)
1219 E09A8 0000             CON(5)   =HERE      HERE to be left on stack as pointer f
          0
1220                *                                   ENDOF/ENDCASE to fill in their ad
1221 E09AD 0920             CON(5)   =ZERO      0
          E
1222 E09B2 CD20             CON(5)   =COMMA     ,   allot space for addr to be filled
          E
1223 E09B7 0000             CON(5)   =COMPL     COMPILE
          0
1224 E09BC 0000             CON(5)   =DROP      DROP  at runtime drop the index value
          0
1225                *                                    since we matched
1226 E09C1 0000             CON(5)   =LIT
          0
1227 E09C6 5000             CON(5)  5           5  leave 5 for syntax checking by END
          0
1228 E09CB 0000             CON(5)   =SEMI      ;
          0
1229
1230                ****************************************
1231                *
1232                * 4+  (n ---  n+4)
1233                *
1234                ****************************************
1235
1236 E09D0 5D90 =FOUR+ CON(5)  $four+
          E
1237 E09D5 147  $four+ C=DAT1   A           read n
1238 E09D8 E6          C=C+1    A           n+1
1239 E09DA E6          C=C+1    A           n+2
1240 E09DC E6          C=C+1    A           n+3
1241 E09DE E6          C=C+1    A           n+4
1242 E09E0 145         DAT1=C   A           write n+4
1243 E09E3 03          RTNCC
```

```
1244                    EJECT
1245                *************************************************************
1246                *
1247                * R> ( -- n) transfer n from return stack to data stack
1248                *
1249                *************************************************************
1250
1251 E09E5 2390             CON(5)  =!OF
          E
1252 E09EA 28    =!R>      CON(2)  #82             R>
1253 E09EC 25               CON(2)  \R\
1254 E09EE EB               CON(2)  \>\+#80
1255 E09F0 5F90  =R>       CON(5)  =$R>
          E
1256
1257 E09F5 D9    =$R>      C=B     A               RTN STK TO C
1258 E09F7 137              CD1EX                   D1=RTN STK,  C=DATA STK
1259 E09FA 143              A=DAT1  A               A=n FROM RTN STK
1260 E09FD 174              D1=D1+  5
1261 E0A00 137              CD1EX                   D1=DATA STK, C=RTN STK
1262 E0A03 D5               B=C     A               RESTORE RTN STK TO B
1263 E0A05 1C4              D1=D1-  5
1264 E0A08 141              DAT1=A  A               PUSH VALUE TO DATA STK
1265 E0A0B 03               RTNCC
```



```
1266                    EJECT
1267                *************************************************************
1268                *
1269                * >R  (n -- )  transfer n to return stack from data stack
1270                *
1271                *************************************************************
1272
1273 E0A0D AE90             CON(5)  !R>
          E
1274 E0A12 28    =!>R      CON(2)  #82             >R
1275 E0A14 E3               CON(2)  \>\
1276 E0A16 2D               CON(2)  \R\+#80
1277 E0A18 D1A0  =>R       CON(5)  =$>R
          E
1278
1279 E0A1D D9    =$>R      C=B     A               RTN STK TO C
1280 E0A1F 143              A=DAT1  A               POP N TO A
1281 E0A22 174              D1=D1+  5
1282 E0A25 137              CD1EX                   DATA STK TO C, RTN STK TO D1
1283 E0A28 1C4              D1=D1-  5
1284 E0A2B 141              DAT1=A  A               PUSH N TO RTN STK
1285 E0A2E 137              CD1EX                   D1=DATA STK, C=RTN STK
1286 E0A31 D5               B=C     A               RESTORE RTN STK
1287 E0A33 03               RTNCC
```

```
1288                    EJECT
1289              ********************************************************
1290              *
1291              * R@  (  --  n)  copy top value from rtn stk to data stk
1292              *
1293              * FORTH word I does the same thing, except that it is
1294              * IMMEDIATE, compile-only.
1295              *
1296              ********************************************************
1297
1298 E0A35 21A0           CON(5)  =!>R
          E
1299 E0A3A 28    =!R@     CON(2)  #82             R@
1300 E0A3C 25             CON(2)  \R\
1301 E0A3E 0C             CON(2)  \@\+#80
1302 E0A40        =I
1303 E0A40 54A0 =R@       CON(5)  =$R@
          E
1304
1305 E0A45 D9    =$R@     C=B     A               RTN STK TO C
1306 E0A47 137            CD1EX                   C=DATA STK, D1=RTN STK
1307 E0A4A 143            A=DAT1  A               TOP ITEM FROM RTN STK
1308              *                               NOTE: ITS COPIED, NOT POPPED
1309 E0A4D 135            D1=C                    D1=DATA STK
1310 E0A50 1C4            D1=D1-  5
1311 E0A53 141            DAT1=A  A               PUSH ITEM TO DATA STK
1312 E0A56 03             RTNCC
```



```
1313                    EJECT
1314              **********************
1315              *
1316              * OR  (n1 n2 --- n3)
1317              *
1318              **********************
1319
1320 E0A58 A3A0           CON(5)  !R@
          E
1321 E0A5D 28    =!OR     CON(2)  #82             OR
1322 E0A5F F4             CON(2)  \O\
1323 E0A61 2D             CON(2)  \R\+#80
1324 E0A63 86A0 =OR       CON(5)  =$OR
          E
1325
1326 E0A68 147   =$OR     C=DAT1  A               POP n2
1327 E0A6B 174            D1=D1+  5
1328 E0A6E 143            A=DAT1  A               POP n1
1329 E0A71 0EFA           C=C!A   A               n1 OR n2
1330 E0A75 145            DAT1=C  A               PUSH RESULT
1331 E0A78 03             RTNCC
1332
1333              ********************************************************
1334              *
1335              * 0=  (n1  ---  t/f)    Returns 0 if n1 <> 0 (false)
1336              *                       else it returns -1 (true)
1337              *
1338              ********************************************************
1339
1340 E0A7A D5A0           CON(5)  =!OR
          E
1341 E0A7F 28    =!ZEQ    CON(2)  #82             0=
1342 E0A81 03             NIBASC  \0\
1343 E0A83 DB             CON(2)  \=\+#80
1344 E0A85 A8A0 =ZEQ      CON(5)  =$ZEQ
          E
1345
1346 E0A8A D0    =$ZEQ    A=0     A               PREPARE FALSE FLAG
1347 E0A8C 147            C=DAT1  A               POP FLAG
1348 E0A8F 8AE            ?C#0    A               NON-ZERO?
1349 E0A92 40             GOYES   ZEQ0            YES, RETURN FALSE FLAG
1350 E0A94 CC             A=A-1   A               MAKE IT ALL 1'S FOR TRUE
1351 E0A96 141  ZEQ0      DAT1=A  A               PUSH FLAG2
1352 E0A99 03             RTNCC
```

```
1353                    EJECT
1354                ***********************************************
1355                *
1356                * +!  (n addr ---  ) add n to value at addr
1357                *
1358                ***********************************************
1359
1360 E0A9B F7A0            CON(5)  !ZEQ
          E
1361 E0AA0 28    =!PSTOR   CON(2)  #82             +!
1362 E0AA2 B2              CON(2)  \+\
1363 E0AA4 1A              CON(2)  \!\+#80
1364 E0AA6 BAA0  =PSTOR    CON(5)  =$PSTOR
          E
1365
1366 E0AAB 147   =$PSTOR   C=DAT1  A                 POP addr
1367 E0AAE 174             D1=D1+  5
1368 E0AB1 143             A=DAT1  A                 POP n
1369 E0AB4 174             D1=D1+  5
1370 E0AB7 137             CD1EX                     DATA STK TO C, D1=ADDR
1371 E0ABA D7              D=C     A                 SAVE DATA STK PTR
1372 E0ABC 147             C=DAT1  A                 CONTENTS OF ADDR
1373 E0ABF C2              C=C+A   A                 ADD
1374 E0AC1 145             DAT1=C  A                 STORE NEW QUANTITY
1375 E0AC4 DB              C=D     A                 RECOVER DATA STK
1376 E0AC6 135             D1=C                      RESTORE DATA STK
1377 E0AC9 03              RTNCC
```



```
1378                    EJECT
1379                ************************
1380                *
1381                * 2*    (n1 --- 2*n1)
1382                *
1383                ************************
1384 E0ACB 0AA0            CON(5)  !PSTOR
          E
1385 E0AD0 28    =!TWO*    CON(2)  #82             2*
1386 E0AD2 23              CON(2)  \2\
1387 E0AD4 AA              CON(2)  \*\+#80
1388 E0AD6 BDA0  =TWO*    CON(5)  =$TWO*
          E
1389
1390 E0ADB 147   =$TWO*   C=DAT1  A                 POP n1
1391 E0ADE C6              C=C+C   A                 n1+n1
1392 E0AE0 145             DAT1=C  A                 PUSH 2*n1
1393 E0AE3 03              RTNCC
1394
1395                ****************************
1396                *
1397                * 2/    (n1 --- n1/2)
1398                *
1399                ****************************
1400
1401 E0AE5 0DA0            CON(5)  =!TWO*
          E
1402 E0AEA 28    =!TWO/    CON(2)  #82             2/
1403 E0AEC 23              CON(2)  \2\
1404 E0AEE FA              CON(2)  \/\+#80
1405 E0AF0 5FA0  =TWO/     CON(5)  =$TWO/
          E
1406
1407 E0AF5 8E00  =$TWO/    GOSUBL  =$ABS           ABS(n1), P=0 IF NEG
          00
1408 E0AFB AD2             C=0     M               CLEAR MANTISSA FOR SHIFT
1409                *                              CAN'T USE P TO CLEAR BECAUSE
1410                *                              ABS USES P AS A NEG FLAG
1411
1412 E0AFE 147             C=DAT1  A               READ ABS(n1)
1413 E0B01 81E             CSRB                    DIVIDE BY 2 (1 BIT SHIFT)
1414 E0B04 880             ?P#     0               WAS # POSITIVE?
1415 E0B07 40              GOYES   T/02           IF YES WE'RE DONE
1416 E0B09 FA              C=-C    A               RESTORE SIGN
1417 E0B0B 145   T/02      DAT1=C  A
1418 E0B0E 03              RTNCC
```

```
1419                    EJECT
1420              ************************************************************
1421              *
1422              * 5+     (n --- n+5)  handy for 20 bit address manipulations
1423              *
1424              ************************************************************
1425
1426 E0B10 AEA0           CON(5)  =!TWO/
         E
1427 E0B15 28    =!FIVE+  CON(2)  #82             5+
1428 E0B17 53             CON(2)  \5\
1429 E0B19 BA             CON(2)  \+\+#80
1430 E0B1B 02B0 =FIVE+    CON(5)  =$FIVE+
         E
1431
1432 E0B20 143   =$FIVE+  A=DAT1  A               POP n
1433 E0B23 20             P=      0
1434 E0B25 3450           LC(5)   5               C(A)=5
         000
1435 E0B2C CA    FV0      A=A+C   A               n+5
1436 E0B2E 141            DAT1=A  A               PUSH n+5
1437 E0B31 03             RTNCC
1438
1439              ************************************************************
1440              *
1441              * FIVE-  (n --- n-5) handy for 20 bit addr manipulations
1442              *
1443              ************************************************************
1444
1445 E0B33 51B0           CON(5)  !FIVE+
         E
1446 E0B38 28    =!FIVE-  CON(2)  #82             5-
1447 E0B3A 53             CON(2)  \5\
1448 E0B3C DA             CON(2)  \-\+#80
1449 E0B3E 34B0 =FIVE-    CON(5)  =$FIVE-
         E
1450
1451 E0B43 143   =$FIVE-  A=DAT1  A               POP n
1452 E0B46 20             P=      0
1453 E0B48 34BF           LC(5)   #FFFFB          2'S COMP OF -5
         FFF
1454 E0B4F 6CDF           GOTO    FV0             FINISH IN FIVE+
```



```
1455                    EJECT
1456
1457              ***************************
1458              *
1459              * TWO+   (n1 --- n1+2)
1460              *
1461              ***************************
1462
1463 E0B53 83B0           CON(5)  !FIVE-
         E
1464 E0B58 28    =!TWO+   CON(2)  #82             2+
1465 E0B5A 23             CON(2)  \2\
1466 E0B5C BA             CON(2)  \+\+#80
1467 E0B5E 36B0 =TWO+     CON(5)  =$TWO+
         E
1468
1469 E0B63 147   =$TWO+   C=DAT1 A               READ n1
1470 E0B66 E6             C=C+1   A               n1=n1+1
1471 E0B68 E6             C=C+1   A               n1=n1+2
1472 E0B6A 145            DAT1=C  A               WRITE n1+2
1473 E0B6D 03             RTNCC
1474
1475              ************************
1476              *
1477              * ONE+   (n1 --- n1+1)
1478              *
1479              ************************
1480
1481 E0B6F 85B0           CON(5)  =!TWO+
         E
1482 E0B74 28    =!ONE+   CON(2)  #82             1+
1483 E0B76 13             CON(2)  \1\
1484 E0B78 BA             CON(2)  \+\+#80
1485 E0B7A F7B0 =ONE+     CON(5)  =$ONE+
         E
1486
1487 E0B7F 147   =$ONE+   C=DAT1  A               POP n1
1488 E0B82 E6             C=C+1   A               INCREMENT n1 BY 1
1489 E0B84 145            DAT1=C  A               PUSH n1+1
1490 E0B87 03             RTNCC
```

```
1491                    EJECT
1492                ***************************
1493                *
1494                * TWO-  (n1 --- n1-2)
1495                *
1496                ***************************
1497
1498 E0B89 47B0           CON(5)  =!ONE+
          E
1499 E0B8E 28     =!TWO-  CON(2)  #82             2-
1500 E0B90 23             CON(2)  \2\
1501 E0B92 DA             CON(2)  \-\+#80
1502 E0B94 99B0   =TWO-   CON(5)  =$TWO-
          E
1503
1504 E0B99 147    =$TWO-  C=DAT1 A                READ n1
1505 E0B9C CE             C=C-1  A                n1=n1-1
1506 E0B9E CE             C=C-1  A                n1=n1-2
1507 E0BA0 145            DAT1=C A                WRITE n1-2
1508 E0BA3 03             RTNCC
1509
1510                ********************************
1511                *
1512                * ONE-  (n1 --- n1-1)
1513                *
1514                ********************************
1515
1516 E0BA5 E8B0           CON(5)  =!TWO-
          E
1517 E0BAA 28     =!ONE-  CON(2)  #82             1-
1518 E0BAC 13             CON(2)  \1\
1519 E0BAE DA             CON(2)  \-\+#80
1520 E0BB0 5BB0   =ONE-   CON(5)  =$ONE-
          E
1521
1522 E0BB5 147    =$ONE-  C=DAT1  A               POP n1
1523 E0BB8 CE             C=C-1   A               DECREMENT n1 BY 1
1524 E0BBA 145            DAT1=C  A               PUSH n1-1
1525 E0BBD 03             RTNCC
```



```
1526                    EJECT
1527                ************************************************
1528                *
1529                * 0<  (n --- flag)
1530                *
1531                * Return true if n is negative else return false
1532                *
1533                ************************************************
1534
1535 E0BBF AAB0           CON(5)  =!ONE-
          E
1536 E0BC4 28     =!LTZ   CON(2)  #82             0<
1537 E0BC6 03             CON(2)  \0\
1538 E0BC8 CB             CON(2)  \<\+#80
1539 E0BCA FCB0   =LTZ    CON(5)  =$LTZ
          E
1540
1541 E0BCF 147    =$LTZ   C=DAT1  A               POP n
1542 E0BD2 D0             A=0     A               PREPARE FOR FALSE
1543 E0BD4 C6             C=C+C   A               CARRY IF NEGATIVE
1544 E0BD6 540            GONC    LTZ0            RETURN FALSE IF POSITIVE
1545 E0BD9 CC             A=A-1   A               NO, TRUE FLAG (ALL 1'S)
1546 E0BDB 141    LTZ0    DAT1=A  A               PUSH FLAG
1547 E0BDE 03             RTNCC
```

```
1548                    EJECT
1549                *************************************
1550                *
1551                * 0>  (n --- flag)
1552                *
1553                * Return true if n > 0 else false
1554                *
1555                *************************************
1556
1557 E0BE0 4CB0             CON(5)  =!LTZ
              E
1558 E0BE5 28    =!GTZ      CON(2)  #82             0>
1559 E0BE7 03               CON(2)  \0\
1560 E0BE9 EB               CON(2)  \>\+#80
1561 E0BEB 0FB0  =GTZ       CON(5)  =$GTZ
              E
1562
1563 E0BF0 147   =$GTZ      C=DAT1  A               POP n
1564 E0BF3 D0               A=0     A               PREPARE FALSE FLAG
1565 E0BF5 8AA              ?C=0    A               IF N=0 THEN FALSE
1566 E0BF8 90               GOYES   GTZ0            FALSE RETURN 0
1567 E0BFA C6               C=C+C   A               CARRY IF NEGATIVE
1568 E0BFC 440              GOC     GTZ0            NEGATIVE RETURN FALSE
1569 E0BFF CC               A=A-1   A               NO, TRUE FLAG (ALL 1'S)
1570 E0C01 141   GTZ0       DAT1=A  A               PUSH FLAG
1571 E0C04 03               RTNCC
```



```
1572                    EJECT
1573                *************************************************
1574                *
1575                * U< (un1 un2 --- flag)  unsigned compare
1576                *
1577                *************************************************
1578
1579 E0C06 5EB0             CON(5)  =!GTZ
              E
1580 E0C0B 28    =!U<       CON(2)  #82             U<
1581 E0C0D 55               CON(2)  \U\
1582 E0C0F CB               CON(2)  \<\+#80
1583 E0C11 61C0  =U<       CON(5)  =$U<
              E
1584
1585 E0C16 147   =$U<      C=DAT1  A               POP un2
1586 E0C19 174              D1=D1+  5
1587 E0C1C 143              A=DAT1  A               POP un1
1588 E0C1F 8B2              ?A<C    A               un1 < un2?
1589 E0C22 90               GOYES   U<TR            YES
1590 E0C24 D0               A=0     A               FALSE FLAG
1591 E0C26 141              DAT1=A  A               STORE IT
1592 E0C29 03               RTNCC
1593 E0C2B D0    U<TR       A=0     A
1594 E0C2D CC               A=A-1   A               TRUE = ALL 1'S
1595 E0C2F 141              DAT1=A  A               WRITE FLAG
1596 E0C32 03               RTNCC
```

```
1597                    EJECT
1598              ***************************************************
1599              *
1600              * D- (d1 d2 --- d3) leave diff of 2 dbl #'s on stack
1601              *
1602              ***************************************************
1603
1604 E0C34 B0C0          CON(5)  =!U<
          E
1605 E0C39 28    =!D-    CON(2)  #82             D-
1606 E0C3B 44            CON(2)  \D\
1607 E0C3D DA            CON(2)  \-\+#80
1608 E0C3F 44C0  =D-      CON(5)  =$D-
          E
1609
1610 E0C44 8E00  =$D-     GOSUBL  =$DNGAT           MAKE D2= -D2
          00
1611 E0C4A 6310          GOTO    $D+
1612
1613              ***********************************************************
1614              *
1615              * D+  (d1 d2 --- d3) leave sum of 2 double numbers on stack
1616              *
1617              ***********************************************************
1618
1619 E0C4E 93C0          CON(5)  =!D-
          E
1620 E0C53 28    =!D+    CON(2)  #82
1621 E0C55 44            CON(2)  \D\
1622 E0C57 BA            CON(2)  \+\+#80
1623 E0C59 E5C0  =D+      CON(5)  =$D+
          E
1624
1625 E0C5E 1C4   =$D+     D1=D1-  5
1626 E0C61 15F9          C=DAT1  10              get MSDs of d2
1627 E0C65 179           D1=D1+  10
1628 E0C68 147           C=DAT1  A               get LSDs of d2
1629 E0C6B 15B9          A=DAT1  10              get MSDs of d1
1630 E0C6F 179           D1=D1+  10
1631 E0C72 143           A=DAT1  A               get LSDs of d1
1632 E0C75 29            P=      9
1633 E0C77 A12           C=C+A   WP              ADD
1634 E0C7A 145           DAT1=C  A               put LSDs of d3
1635 E0C7D 1C9           D1=D1-  10
1636 E0C80 15D9          DAT1=C  10              put MSDs of d3
1637 E0C84 174           D1=D1+  5
1638 E0C87 03            RTNCC
```



```
1639                    EJECT
1640              ***************************************************
1641              *
1642              * BL  ( -- " ") Push an ASCII blank on the DATA STK
1643              *
1644              ***************************************************
1645
1646 E0C89 35C0          CON(5)  =!D+
          E
1647 E0C8E 28    =!BL    CON(2)  #82             BL
1648 E0C90 24            CON(2)  \B\
1649 E0C92 CC            CON(2)  \L\+#80
1650 E0C94 0000  =BL      CON(5)  =DOCON
          0
1651 E0C99 0200          CON(5)  \ \
          0
```

```
1652                    EJECT
1653              ********************************************
1654              *
1655              * C@  (addr -- b)  Fetch byte from given addr.
1656              *                  NOTE: it puts 5 NIBS on the
1657              *                  DATA STK, the high order NIBS
1658              *                  ARE always 0
1659              *
1660              ********************************************
1661
1662 E0C9E E8C0           CON(5)  =!BL
          E
1663 E0CA3 28   =!CAT     CON(2)  #82             C@
1664 E0CA5 34             CON(2)  \C\
1665 E0CA7 0C             CON(2)  \@\+#80
1666 E0CA9 EAC0 =CAT      CON(5)  =$CAT
          E
1667
1668 E0CAE 147  =$CAT     C=DAT1  A       pop addr
1669 E0CB1 137            CD1EX           D1=addr, C=DATA STK
1670 E0CB4 D0             A=0     A       clear high nibbles
1671 E0CB6 14B            A=DAT1  B       read byte
1672 E0CB9 135            D1=C            D1=DATA STK
1673 E0CBC 141            DAT1=A  A       push byte
1674 E0CBF 03             RTNCC
1675
1676              ****************************************
1677              *
1678              * C!  (c addr --- ) store byte at addr.
1679              *
1680              ****************************************
1681
1682 E0CC1 3AC0           CON(5)  =!CAT
          E
1683 E0CC6 28   =!C!      CON(2)  #82
1684 E0CC8 34             NIBASC  \C\
1685 E0CCA 1A             CON(2)  \!\+#80
1686 E0CCC 1DC0 =C!       CON(5)  =$C!
          E
1687
1688 E0CD1 147  =$C!      C=DAT1  A       POP ADDR
1689 E0CD4 174            D1=D1+  5
1690 E0CD7 143            A=DAT1  A       POP c
1691 E0CDA 174            D1=D1+  5       RETURN NOTHING ON STACK
1692 E0CDD 137            CD1EX           ADDR TO D1, DATA STK TO C
1693 E0CE0 149            DAT1=A  B       STORE BYTE AT ADDR
1694 E0CE3 135            D1=C            RESTORE DATA STK TO C
1695 E0CE6 03             RTNCC
```



```
1696                    EJECT
1697              ********************************************************
1698              *
1699              * DOUSE    DOUSE is used to put the address of a User
1700              *          variable on the DATA STACK.
1701              *
1702              *         <USER VAR HEADER>
1703              *         <gosub DOUSE>
1704              *         <addr of User Var
1705              *          in User Var Table>
1706              *
1707              *
1708              ********************************************************
1709 E0CE8 07   =DOUSE   C=RSTK          POP ADDR OF OFFSET INTO UVAR
1710 E0CEA 136          CD0EX
1711 E0CED 142          A=DAT0  A       POP ADDR OF UVAR TO A(A)
1712 E0CF0 134          D0=C            RESTORE I TO D0
1713 E0CF3 1C4          D1=D1-  5
1714 E0CF6 141          DAT1=A  A       push addr to DATA STACK
1715 E0CF9 03           RTNCC
```

```
1716                    EJECT
1717                ***********************************
1718                *
1719                * DP ( --- addr)  addr of next free
1720                *       byte in user dictionary
1721                *
1722                ***********************************
1723
1724 E0CFB 00D0 =DP     CON(5)  =$DP
          E
1725 E0D00 74EF =$DP    GOSUB   DOUSE
1726 E0D04 39BF         CON(5)   (=oDP)
          2
```



```
1727                    EJECT
1728                ************************************************
1729                *
1730                * IF    compile time: ( - addr 2 )     immediate
1731                *
1732                * Puts ZBRNH code in word with a zero addr to be filled
1733                * in by THEN. It also leaves a 2 on the stack which
1734                * THEN knows to check for, just as a signal that
1735                * things match up.
1736                *
1737                ************************************************
1738
1739 E0D09 6CC0         CON(5)  =!C!
          E
1740 E0D0E 2C   =!IF    CON(2) #C2            IF
1741 E0D10 94           NIBASC \I\
1742 E0D12 6C           CON(2) \F\+#80
1743
1744 E0D14 0000 =IF     CON(5) =DOCOL
          0
1745 E0D19 0000         CON(5)  =COMPL        COMPILE
          0
1746 E0D1E 0000         CON(5)  =ZBRNH        0BRANCH
          0
1747 E0D23 0000         CON(5)  =HERE         leave pointer to addr to be filled
          0
1748                *                         in by THEN on stack
1749 E0D28 0920         CON(5)  =ZERO          0
          E
1750 E0D2D CD20         CON(5)  =COMMA       allot space for addr filled in later
          E
1751 E0D32 6B20         CON(5)  =TWO           2    for syntax checking by THEN
          E
1752 E0D37 0000         CON(5)  =SEMI          ;
          0
```

```
1753                    EJECT
1754              ******************************************
1755              *
1756              * DO   compile time: ( - addr 3 )   immediate
1757              *
1758              * compiles XDO
1759              *
1760              * The 3 left on stack is there to be checked by
1761              * LOOP, and to cause an error if there isn't
1762              * any LOOP.
1763              *
1764              ******************************************
1765
1766 E0D3C E0D0           CON(5) =!IF
          E
1767 E0D41 2C    =!DO     CON(2) #C2              IMMEDIATE
1768 E0D43 44             NIBASC \D\
1769 E0D45 FC             CON(2) \O\+#80
1770
1771 E0D47 0000 =DO       CON(5) =DOCOL
          0
1772 E0D4C 0000           CON(5)  =COMPL          COMPILE
          0
1773 E0D51 0000           CON(5)  =XDO            XDO
          0
1774 E0D56 0000           CON(5)  =HERE           leave ptr to beginning of
          0
1775              *                               loop for LOOP to branch back to
1776 E0D5B 9C20           CON(5)  =THREE          3  syntax check for LOOP
          E
1777 E0D60 0000           CON(5)  =SEMI           ;
          0
```



```
1778                    EJECT
1779              ******************************************************
1780              *
1781              * ."  (DOT-QUOTE)  compile the text up to but not including
1782              *     the delimiting "   When later executed display
1783              *     the text on the active display device.
1784              *
1785              *     According to the FORTH 83 Standard this word is
1786              *     legal ONLY during compile mode.  .( (dot paren) has
1787              *     taken the place of ."  in the execute mode.
1788              *
1789              *     IMMEDIATE
1790              *
1791              ******************************************************
1792
1793 E0D65 14D0           CON(5)  =!DO
          E
1794 E0D6A 2C    =!DOTQ   CON(2)  #C2
1795 E0D6C E2             NIBASC  \.\
1796 E0D6E 2A             CON(2)  \"\+#80
1797
1798 E0D70 0000 =DOTQ     CON(5) =DOCOL
          0
1799 E0D75 0000           CON(5)  =?COMP          enforce compile mode.
          0
1800 E0D7A 0000           CON(5)   =STRM          'STREAM   find source
          0
1801 E0D7F 9AC0           CON(5)   =CAT           C@  read next char
          E
1802 E0D84 0000           CON(5)   =LIT
          0
1803 E0D89 2200           CON(5)  34              \"\  see if its a quote
          0
1804 E0D8E 8C10           CON(5)   =EQUAL         =   if so
          E
1805 E0D93 0000           CON(5)   =ZBRNH         IF   we have a null string
          0
1806 E0D98 F000           CON(5)  (DO)-*               so just add
          0
1807 E0D9D 0000           CON(5)   =PT1+          2 >IN +!  adds 2 to input counter
          0
1808 E0DA2 0000           CON(5)   =SEMI          cheat and end here
          0
1809
1810 E0DA7 0000 DO        CON(5)   =LIT                 of the string
          0
1811 E0DAC 2200           CON(5)  34              \"\   delimited by a quote
          0
1812 E0DB1 0000           CON(5)   =COMPL         COMPILE   compile the cfa of
          0
1813 E0DB6 AFE0           CON(5)   =PDOTQ         <.">    runtime code for ."
          E
1814 E0DBB 0000           CON(5)   =GCHAR         move string to HERE, and make
          0
1815              *                               sure there's an ending "
```

```
1816                *                                  returns 1 if no ending "
1817 E0DC0 0000            CON(5)   =ABRT"X
           0
1818 E0DC5 50              CON(2)   =eNO"
1819                *      NIBASC   \no endin\
1820                *      NIBASC   \g  "\
1821 E0DC7 0000            CON(5)   =MOVDP      C@ 1+ 2* ALLOT
           0
1822 E0DCC 0000 D03        CON(5)   =SEMI        THEN THEN ;
           0
```



```
1823                EJECT
1824              ************************************************************
1825              *
1826              * .(  (DOT-PAREN)  The characters up to but not including
1827              *      the delimiting ) (closing paren) are displayed
1828              *      regardless of mode.   FORTH 83 Standard
1829              *
1830              *      IMMEDIATE
1831              *
1832              ************************************************************
1833
1834 E0DD1 A6D0            CON(5)   =!DOTQ
           E
1835 E0DD6 2C    =!DOTP    CON(2)   #C2
1836 E0DD8 E2              NIBASC   \.\
1837 E0DDA 8A              CON(2)   \(\+#80
1838
1839 E0DDC 0000 =DOTP      CON(5)  =DOCOL
           0
1840 E0DE1 0000            CON(5)    =STRM       'STREAM      find addr of input stre
           0
1841 E0DE6 9AC0            CON(5)    =CAT        C@           get 1st char
           E
1842 E0DEB 0000            CON(5)    =LIT
           0
1843 E0DF0 9200            CON(5)   41           \)\          see if its the delimiti
           0
1844 E0DF5 8C10            CON(5)    =EQUAL      =            paren,
           E
1845 E0DFA 0000            CON(5)    =ZBRNH      IF           if so we have a null
           0
1846 E0DFF F000            CON(5)   (PD0)-*                   string so
           0
1847 E0E04 0000            CON(5)    =PT1+       2 >IN +!     just add 2 to the input
           0
1848                *                                         nibble counter
1849 E0E09 0000            CON(5)    =SEMI       cheat and end here!
           0
1850
1851 E0E0E 0000 PD0        CON(5)    =LIT                     look for end of string
           0
1852 E0E13 9200            CON(5)   41           \)\          delimited by ending par
           0
1853 E0E18 0000            CON(5)    =GCHAR      move string to here, make sure that
           0
1854                *                             string ends with ), returns 1 if
1855                *                             ending ) is missing! and string add
1856 E0E1D 0000            CON(5)    =ABRT"X
           0
1857 E0E22 40              CON(2)    =eNOEP
1858                *      NIBASC    \no endin\
1859                *      NIBASC    \g  )\
1860 E0E24 0000            CON(5)    =COUNT      COUNT        get string params
           0
1861 E0E29 0000            CON(5)    =TYPE       TYPE         dispaly string
```

```
            0
1862 E0E2E 0000 PD03    CON(5)   =SEMI        THEN ;
            0
```



```
1863                EJECT
1864            ********************************************************
1865            *
1866            * S0 ( --- addr)  Returns addr of bottom of data stack.
1867            *
1868            ********************************************************
1869
1870 E0E33 6DD0         CON(5)   =!DOTP
            E
1871 E0E38 28   =!S0    CON(2)   #82             S0
1872 E0E3A 35           CON(2)   \S\
1873 E0E3C 0B           CON(2)   \0\+#80
1874 E0E3E 0000 =S0     CON(5)   =DOCOL          :
            0
1875 E0E43 0000         CON(5)    =SP0           SP0
            0
1876 E0E48 1720         CON(5)    =AT            @
            E
1877 E0E4D 0000         CON(5)    =SEMI          ;
            0
```

```
1878                    EJECT
1879                ****************************************************************
1880                *
1881                * M/  (d n1 --- n2 n3) Mixed magnitude operator, leaves the signe
1882                *       remainder, n2, and signed quotient, n3, the remainder tak
1883                *       its sign from the dividend.
1884                *
1885                ****************************************************************
1886
1887 E0E52 83E0             CON(5)  =!S0
          E
1888 E0E57 28     =!M/      CON(2)  #82             M/
1889 E0E59 D4               CON(2)  \M\
1890 E0E5B FA               CON(2)  \/\+#80
1891
1892 E0E5D 0000   =M/       CON(5)  =DOCOL          :
          0
1893 E0E62 0000             CON(5)   =OVER          d1 n1 d1
          0
1894 E0E67 81A0             CON(5)   =>R            d1 n1  :: d1
          E
1895 E0E6C 81A0             CON(5)   =>R            d1     :: n1 d1
          E
1896 E0E71 0000             CON(5)   =DABS          |d1|
          0
1897 E0E76 04A0             CON(5)   =R@            |d1| n1
          E
1898 E0E7B 0000             CON(5)   =ABS           |d1|  |n1|
          0
1899 E0E80 0000             CON(5)   =UDIV          urem uquot
          0
1900 E0E85 0F90             CON(5)   =R>            urem uquot n1
          E
1901 E0E8A 04A0             CON(5)   =R@            urem uquot n1 (high-nibs of d1)
          E
1902 E0E8F 0000             CON(5)   =XOR           XOR rtn 0 if n1 & high-nibs of sa
          0
1903 E0E94 0000             CON(5)   =?NEG          ?NEGATE  negate uquot if only one
          0
1904                *                               n1 & high-nibs were negative
1905 E0E99 0000             CON(5)   =SWAP          uquot urem
          0
1906 E0E9E 0F90             CON(5)   =R>            uquot urem (low-nibs of d1)
          E
1907 E0EA3 0000             CON(5)   =?NEG          negate urem if low-nibs negative
          0
1908 E0EA8 0000             CON(5)   =SWAP          urem uquot
          0
1909 E0EAD 0000             CON(5)   =SEMI           ;
          0
```



```
1910                    EJECT
1911                ****************************************************
1912                * D. (D-DOT)    (d --- ) Display d converted according to BASE
1913                *               in a free-field format, with one trailing blank.
1914                *               Display sign only if negative.
1915                *
1916                ****************************************************
1917
1918 E0EB2 75E0             CON(5)  =!M/
          E
1919 E0EB7 28     =!D.      CON(2)  #82             D.
1920 E0EB9 44               CON(2)  \D\
1921 E0EBB EA               CON(2)  \.\+#80
1922
1923 E0EBD 0000   =D.       CON(5)  =DOCOL          : D.
          0
1924 E0EC2 0920             CON(5)   =ZERO          0
          E
1925 E0EC7 0000             CON(5)   =D.R           D.R
          0
1926 E0ECC 0000             CON(5)   =SPACE         SPACE
          0
1927 E0ED1 0000             CON(5)   =SEMI          ;
          0
1928                ****************************************************
1929                *
1930                * */ (n1 n2 n3 -- n4)  Multiply n1 by n2 divide result
1931                *       by n3, leave quotient n4.
1932                *
1933                ********************************************
1934
1935 E0ED6 7BE0             CON(5)  =!D.
          E
1936 E0EDB 28     =!*/      CON(2)  #82
1937 E0EDD A2               NIBASC  \*\
1938 E0EDF FA               CON(2)  \/\+#80
1939
1940 E0EE1 0000   =*/       CON(5)  =DOCOL          :
          0
1941 E0EE6 0000             CON(5)   =*/MOD         rem quot
          0
1942 E0EEB 0000             CON(5)   =SWAP          quot rem
          0
1943 E0EF0 0000             CON(5)   =DROP          quot
          0
1944 E0EF5 0000             CON(5)   =SEMI           ;
          0
```

```
1945                    EJECT
1946                ******************************************
1947                *
1948                * ."    SEND A STRING TO THE DISPLAY
1949                *       COMPILED AFTER THE CFA FOR ." IS
1950                *       THE CHAR COUNT AND THE CHARS
1951                *
1952                ******************************************
1953
1954 E0EFA 0000 =PDOTQ  CON(5)   =DOCOL          :
          0
1955 E0EFF 04A0        CON(5)   =R@            R@  GET COPY FROM RTN STK
          E
1956 E0F04 0000        CON(5)   =COUNT         COUNT
          0
1957 E0F09 0000        CON(5)   =DUP           DUP
          0
1958 E0F0E A7B0        CON(5)   =ONE+          1+
          E
1959 E0F13 6DA0        CON(5)   =TWO*          2*  MAKE CHAR COUNT=NIBS
          E
1960 E0F18 0F90        CON(5)   =R>            R>
          E
1961 E0F1D C810        CON(5)   =ADD           +  ADDR OF NEXT RTNE
          E
1962 E0F22 81A0        CON(5)   =>R            >R RTN TO RTN STACK
          E
1963 E0F27 0000        CON(5)   =TYPE          TYPE STRING
          0
1964 E0F2C 0000        CON(5)   =SEMI          ;
          0
```



```
1965                    EJECT
1966                ******************************************************
1967                *
1968                * CR    Send a carriage return, line feed to the display
1969                *       device.
1970                *
1971                ******************************************************
1972 E0F31 BDE0        CON(5)   =!*/
          E
1973 E0F36 28    =!CR  CON(2)   #82
1974 E0F38 34          NIBASC   \C\
1975 E0F3A 2D          CON(2)   \R\+#80
1976 E0F3C 14F0 =CR    CON(5)   =$CR
          E
1977
1978 E0F41 8F00 =$CR   GOSBVL   =SAVEFP         SAVE FORTH POINTERS
          000
1979 E0F48 20          P=       0
1980 E0F4A 8F00        GOSBVL   =CRLFOF         CR LF TO DISPLAY
          000
1981 E0F51 8D00        GOVLNG   =GETFP          RESTORE FORTH POINTERS
          000
```

```
1982                    EJECT
1983                ****************************************************************
1984                *
1985                * #S (SHARP-S)  (ud --- 0 0 ) Convert all digits of an unsigned
1986                *               40 bit number ud, adding each to the pictured
1987                *               numeric output text, until remainder is zero.
1988                *               Use between <# and #>.
1989                *
1990                ****************************************************************
1991
1992 E0F58 63F0             CON(5)  =!CR
           E
1993 E0F5D 28     =!N#S     CON(2)  #82             #S
1994 E0F5F 32               CON(2)  \#\
1995 E0F61 3D               CON(2)  \S\+#80
1996 E0F63 0000   =N#S      CON(5)  =DOCOL          : #S
           0
1997                                                BEGIN
1998 E0F68 7830   N00       CON(5)   =N#            #
           E
1999 E0F6D 0000             CON(5)   =2DUP          2DUP
           0
2000 E0F72 36A0             CON(5)   =OR            OR
           E
2001 E0F77 58A0             CON(5)   =ZEQ           0=
           E
2002 E0F7C 0000             CON(5)   =ZBRNH         UNTIL
           0
2003 E0F81 7EFF             CON(5)  (N00)-*
           F
2004 E0F86 0000             CON(5)   =SEMI          ;
           0
2005
2006                ****************************************************************
2007                *
2008                * #>  (SHARP-GREATER)   (ud --- addr n)  End pictured numeric out
2009                *                conversion.  Drop ud, leaving text addr and char
2010                *               suitable for TYPE.
2011                *
2012                ****************************************************************
2013
2014 E0F8B D5F0             CON(5)  =!N#S
           E
2015 E0F90 28     =!GT#     CON(2)  #82             #>
2016 E0F92 32               CON(2)  \#\
2017 E0F94 EB               CON(2)  \>\+#80
2018
2019 E0F96 0000   =GT#      CON(5)  =DOCOL          : #>
           0
2020 E0F9B 0000             CON(5)   =2DROP         drop ud
           0
2021 E0FA0 0000             CON(5)   =HLD           get addr of last char
           0
2022 E0FA5 1720             CON(5)   =AT            converted
           E
```



```
2023 E0FAA 0000             CON(5)   =PAD           addr of PAD
           0
2024 E0FAF 0000             CON(5)   =OVER          <addr last char> pad <addr last c
           0
2025 E0FB4 AA10             CON(5)   =MINUS         get nibble difference
           E
2026 E0FB9 0FA0             CON(5)   =TWO/          divide by 2 for byte count of cha
           E
2027 E0FBE 0000             CON(5)   =SEMI          ;
           0
2028
2029                ****************************************************************
2030                *
2031                * <#  (LESS-SHARP)    ( --- ) Initialize pictured numeric
2032                *       output.
2033                *
2034                ****************************************************************
2035
2036 E0FC3 09F0             CON(5)  =!GT#
           E
2037 E0FC8 28     =!LT#     CON(2)  #82             <#
2038 E0FCA C3               CON(2)  \<\
2039 E0FCC 3A               CON(2)  \#\+#80
2040
2041 E0FCE 0000   =LT#      CON(5)  =DOCOL          : <#
           0
2042 E0FD3 0000             CON(5)   =PAD           set HLD to contain
           0
2043 E0FD8 0000             CON(5)   =HLD           addr of PAD as starting place
           0
2044 E0FDD C420             CON(5)   =STORE
           E
2045 E0FE2 0000             CON(5)   =SEMI          ;
           0
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER

```
2046                    EJECT
2047              ************************
2048              *
2049              * D<  (d1  d2  --- f)
2050              *
2051              ********************************
2052
2053 E0FE7 8CF0           CON(5)  =!LT#
         E
2054 E0FEC 28    =!D<     CON(2)  #82             D<
2055 E0FEE 44             CON(2)  \D\
2056 E0FF0 CB             CON(2)  \<\+#80
2057
2058 E0FF2 7FF0  =D<      CON(5)  =$D<
         E
2059 E0FF7 1C4   =$D<     D1=D1-  5
2060 E0FFA 15B9           A=DAT1  10          GET N2 MSD IN A[5-9]
2061 E0FFE 179            D1=D1+  10
2062 E1001 143            A=DAT1  A           GET N2 LSD IN A[A]
2063 E1004 15F9           C=DAT1  10          GET N1 MSD IN C[5-9]
2064 E1008 179            D1=D1+  10
2065 E100B 147            C=DAT1  A           GET N1 LSD IN C[A]
2066 E100E 2A             P=      10
2067 E1010 A82            C=0     P
2068 E1013 A80            A=0     P           CLEAR FOR TEST
2069 E1016 A16            C=C+C   WP
2070 E1019 A14            A=A+A   WP          SHIFT HIGH BIT INTO NIB 10
2071 E101C 902            ?C=A    P           SAME SIGN? IF SO, TEST WORKS
2072 E101F 50             GOYES   SAMSIN
2073              *                           (BECAUSE BIGGER NEGATIVE NUMBER
2074              *                           IS SMALLER UNSIGNED NUMBER)
2075 E1021 A9E            ACEX    WP          ELSE EXCHANGE (SO FALSE IF N2 NEG)
2076 E1024 996   SAMSIN   ?A>C    WP          N2 > N1 ?
2077 E1027 90             GOYES   D<TRUE
2078 E1029 D0             A=0     A
2079 E102B 141   D<RES    DAT1=A  A           RETURN FALSE
2080 E102E 03             RTNCC
2081 E1030 D0    D<TRUE   A=0     A
2082 E1032 CC             A=A-1   A           TRUE FLAG
2083 E1034 66FF           GOTO    D<RES
2084
2085              ************************************
2086              *
2087              * M* (MIXED MULTIPLY)  (n1 n2 --- d)
2088              *
2089              ************************************
2090
2091 E1038 CEF0           CON(5)  =!D<
         E
2092 E103D 28    =!M*     CON(2)  #82             M*
2093 E103F D4             CON(2)  \M\
2094 E1041 AA             CON(2)  \*\+#80
2095
2096 E1043 0000  =M*      CON(5)  =DOCOL          :
         0
```



```
2097 E1048 0000           CON(5)  =2DUP           n1 n2 n1 n2
         0
2098 E104D 0000           CON(5)  =XOR            produces neg # if 1 or other is
         0
2099 E1052 81A0           CON(5)  =>R             save flag on rtn stk
         E
2100 E1057 0000           CON(5)  =ABS            n1  |n2|
         0
2101 E105C 0000           CON(5)  =SWAP           |n2| n1
         0
2102 E1061 0000           CON(5)  =ABS            |n2|  |n1|
         0
2103 E1066 0000           CON(5)  =U*             d1
         0
2104 E106B 0F90           CON(5)  =R>             recover flag
         E
2105 E1070 0000           CON(5)  =D?NEG          negate d1 if XOR answer is negat
         0
2106 E1075 0000           CON(5)  =SEMI           ;
         0
```

```
2107                    EJECT
2108            ****************************
2109            *
2110            * <>  ( n1  n2 --- n3)
2111            *
2112            * RETURN TRUE IF n1 # n2 ELSE
2113            * RETURN FALSE FLAG
2114            *
2115            ****************************
2116
2117 E107A D301          CON(5)  =!M*
         E
2118 E107F 28    =!<>    CON(2)  #82             <>
2119 E1081 C3            CON(2)  \<\
2120 E1083 EB            CON(2)  \>\+#80
2121
2122 E1085 A801 =<>      CON(5)  $<>
         E
2123 E108A 143  $<>      A=DAT1  A               read n2
2124 E108D 174           D1=D1+  5
2125 E1090 147           C=DAT1  A               read n1
2126 E1093 E2            C=C-A   A
2127 E1095 8AA           ?C=0    A               false flag
2128 E1098 60            GOYES   <>0
2129 E109A D2            C=0     A
2130 E109C CE            C=C-1   A               true flag
2131 E109E 145  <>0      DAT1=C  A               write flag
2132 E10A1 03            RTNCC
2133
2134            ************************************************************
2135            *
2136            * U.  (un --- )  Display un converted according to BASE as an
2137            *       unsigned number.
2138            *
2139            ************************************************************
2140
2141 E10A3 F701          CON(5)  =!<>
         E
2142 E10A8 28    =!U.    CON(2)  #82             U.
2143 E10AA 55            CON(2)  \U\
2144 E10AC EA            CON(2)  \.\+#80
2145
2146 E10AE 0000 =U.      CON(5)  =DOCOL          :
         0
2147 E10B3 0920          CON(5)   =ZERO          0
         E
2148 E10B8 DBE0          CON(5)   =D.            D.
         E
2149 E10BD 0000          CON(5)   =SEMI          ;
         0
```



```
2150                    EJECT
2151            ************************************************************
2152            *
2153            * C,  (n --- ) Store lower order 8 bits of n at next byte
2154            *       in dictionary, advancing dictionary pointer
2155            *
2156            * -----VECTORED WORD-----
2157            *
2158            ************************************************************
2159
2160 E10C2 8A01          CON(5)  =!U.
         E
2161 E10C7 28    =!C,    CON(2)  #82             C,
2162 E10C9 34            CON(2)  \C\
2163 E10CB CA            CON(2)  \,\+#80
2164
2165 E10CD 0000 =C,      CON(5)  =DOCOL          :
         0
2166 E10D2 0000          CON(5)  =LIT
         0
2167 E10D7 43CF          CON(5)  =oCCOMA         VECTOR FOR C,
         2
2168 E10DC 1720          CON(5)  =AT             CFA TO USE
         E
2169 E10E1 0000          CON(5)  =EXEC           EXECUTE IT
         0
2170 E10E6 0000          CON(5)  =SEMI
         0
2171
2172            *
2173            * DEFAULT C, ROUTINE
2174            *
2175
2176 E10EB 0000 =ICCOMA  CON(5)  =DOCOL          :
         0
2177 E10F0 0000          CON(5)   =HERE          addr of first free nib in diction
         0
2178 E10F5 CCC0          CON(5)   =C!            store byte there
         E
2179 E10FA 6B20          CON(5)   =TWO           2  (ALLOT WANTS NIBS)
         E
2180 E10FF 0000          CON(5)   =ALLOT         allot 2 nibs
         0
2181 E1104 0000          CON(5)   =SEMI          ;
         0
```

```
2182                    EJECT
2183              ********************************************
2184              *
2185              * S!                          ( str1 str2 -- )
2186              * Set value of string. Str1 may be a temporary
2187              * created by " .
2188              * If len(str1)>maxlen(str2), give error msg.
2189              *
2190              ********************************************
2191
2192 E1109 7C01          CON(5)  =!C,
          E
2193 E110E 28    =!S!    CON(2)  #82
2194 E1110 35            NIBASC  \S\
2195 E1112 1A            CON(2)  \!\+#80
2196 E1114 0000 =S!      CON(5)  =DOCOL        :
          0
2197 E1119 0000          CON(5)   =2DUP        COPY STR2
          0
2198 E111E 0000          CON(5)   =MXLEN       addr1 len1 addr2 len2 maxlen
          0
2199 E1123 0000          CON(5)   =SWAP        addr1 len1 addr2 maxlen len2
          0
2200 E1128 0000          CON(5)   =DROP        addr1 len1 addr2 maxlen
          0
2201
2202              *  addr1 len1 addr2 max2
2203
2204 E112D 9C20          CON(5)   =THREE
          E
2205 E1132 0000          CON(5)   =PICK
          0
2206              *  addr1 len1 addr2 max2 len1
2207 E1137 FE10           CON(5)  =LT             LEN1>MAX2?
          E
2208 E113C 0000           CON(5)  =ABORTW
          0
2209 E1141 53             CON(2)  =eSWF           "STRING WON'T FIT"
2210              *  addr1 len1 addr2
2211 E1143 0000          CON(5)   =2DUP        2DUP
          0
2212 E1148 49B0          CON(5)   =TWO-        2-
          E
2213 E114D CCC0          CON(5)   =C!          C!
          E
2214              * STORE LEN1 AS NEW CURLEN2
2215              * addr1 len1 addr2
2216 E1152 0000          CON(5)   =SWAP        addr1 addr2 len1
          0
2217 E1157 0000          CON(5)   =CMOVE       move string to addr2
          0
2218 E115C 0000          CON(5)   =SEMI        ;
          0
```



```
2219                    EJECT
2220              ************************************************
2221              *
2222              * S=                          ( str1 str2 -- t/f )
2223              * Tells whether two strings are equal, i.e. of
2224              * same length and having all same characters in
2225              * same order.
2226              *
2227              ************************************************
2228
2229 E1161 E011          CON(5)  =!S!
          E
2230 E1166 28    =!S=    CON(2)  #82
2231 E1168 35            NIBASC  \S\
2232 E116A DB            CON(2)  \=\+#80
2233 E116C 1711 =S=      CON(5)  =$S=
          E
2234
2235 E1171 8F00 =$S=     GOSBVL  =SAVEFP
          000
2236 E1178 7620          GOSUB   SETSTR        put lengths in scratch registers
2237 E117C 118           C=R0                  get len2
2238 E117F 112           A=R2                  len1
2239 E1182 8A6           ?C#A    A             strings of same length?
2240 E1185 51            GOYES   fail          if not return answer of false
2241 E1187 D7            D=C     A             D=len
2242 E1189 7830          GOSUB   RESET         D0&D1 point to strings
2243 E118D 7240          GOSUB   SAME          set carry if they are identical
2244 E1191 4C0           GOC     less          "less" will rtn true flag
2245 E1194 D1            B=0     A             sett B as false flag
2246 E1196 65C0          GOTO    MATCH         rtn B,= false
2247              *
2248 E119A 67C0 fail     GOTO    FAIL
2249 E119E 69B0 less     GOTO    LESS
2250
2251              * subroutines of S= , POS and S<
2252
2253 E11A2 147  =SETSTR C=DAT1  A
2254 E11A5 108          R0=C                  R0=len2
2255 E11A8 174          D1=D1+  5
2256 E11AB 147          C=DAT1  A
2257 E11AE 109          R1=C                  R1=addr2
2258 E11B1 174          D1=D1+  5
2259 E11B4 147          C=DAT1  A
2260 E11B7 10A          R2=C                  R2=len1
2261 E11BA 174          D1=D1+  5
2262 E11BD 147          C=DAT1  A
2263 E11C0 10B          R3=C                  R3=addr1
2264 E11C3 01           RTN
2265
2266 E11C5 119  =RESET C=R1
2267 E11C8 137          CD1EX
2268 E11CB 11B          C=R3
2269 E11CE 136          CD0EX
2270 E11D1 01           RTN
```

```
2271
2272            * TEST FOR EQUALITY, GIVEN: D1,D0 point to
2273            * two strings, D[A]=length of them both.
2274            *
2275
2276 E11D3 8AB  =SAME  ?D=0    A           done with strings?
2277 E11D6 B1          GOYES   SAMEY       (null strings match by definition)
2278 E11D8 14F         C=DAT1  B           get char
2279 E11DB 14A         A=DAT0  B           get char
2280 E11DE 966         ?C#A    B
2281 E11E1 E0          GOYES   SAMENO      some char didn't match
2282 E11E3 161         D0=D0+  2
2283 E11E6 171         D1=D1+  2           point to next chars
2284 E11E9 CF          D=D-1   A           count down length
2285 E11EB 67EF        GOTO    SAME
2286 E11EF 03  SAMENO  RTNCC
2287 E11F1 02  SAMEY   RTNSC
```



```
2288                EJECT
2289            ************************************************
2290            *
2291            * S<                          ( str1 str2 -- t/f )
2292            * Test for str1<str2 (defined in terms of ASCII
2293            * values). A shorter string is < a longer one that
2294            * starts out the same.
2295            *
2296            ************************************************
2297
2298 E11F3 6611        CON(5)  =!S=
                E
2299 E11F8 28  =!S<    CON(2)  #82
2300 E11FA 35          NIBASC  \S\
2301 E11FC CB          CON(2)  \<\+#80
2302 E11FE 3021 =S<    CON(5)  =$S<
                E
2303
2304 E1203 8F00 =$S<   GOSBVL  =SAVEFP
                000
2305 E120A 143         A=DAT1  A           A(A)=len2
2306 E120D 174         D1=D1+  5
2307 E1210 147         C=DAT1  A           get addr2
2308 E1213 136         CD0EX               D0--->addr2
2309 E1216 174         D1=D1+  5
2310 E1219 147         C=DAT1  A           C(A)=len1
2311 E121C 174         D1=D1+  5
2312 E121F D8          B=A     A           assume len1 > len2
2313 E1221 841         ST=0    1           flag for this condition
2314 E1224 8BA         ?A<=C   A           len2 <= len1?
2315 E1227 70          GOYES   S<01
2316 E1229 D5          B=C     A           len2 > len1
2317            * B is set to whichever length is less
2318 E122B 851         ST=1    1           flag len2 > len1
2319            * what's happening here is that if the first string is
2320            * longer, we need to remember that,in case it matches
2321            * up through the end of the second string (at which
2322            * point we say false).
2323 E122E 147  S<01    C=DAT1  A
2324 E1231 137         CD1EX               D1--->addr1
2325 E1234 CD   S<00    B=B-1   A           decrement counter
2326 E1236 4C1         GOC     S<02        done!
2327 E1239 14B         A=DAT1  B           str1
2328 E123C 14E         C=DAT0  B           str2
2329 E123F 9E2         ?A<C    B
2330 E1242 61          GOYES   LESS        return true
2331 E1244 9E6         ?A>C    B
2332 E1247 B1          GOYES   FAIL        return false
2333            * otherwise they are still identical
2334 E1249 171         D1=D1+  2           increment str1
2335 E124C 161         D0=D0+  2           increment str2
2336 E124F 64EF        GOTO    S<00        do another
2337
2338            * strings matched through end of the shorter one
2339 E1253 861  S<02    ?ST=0   1           len1>=len2?
```

```
2340 E1256 C0          GOYES   FAIL
2341
2342 E1258 D1   LESS   B=0     A
2343 E125A CD          B=B-1   A
2344 E125C 8C00 MATCH  GOLONG  =fMATCH
           00
2345 E1262 8C00 FAIL   GOLONG  =fFAIL
           00
2346
2347 E1268      =REND1 END
```



```
=!*/     Abs  921307 #E0EDB -  1936  1972
=!/      Abs  918306 #E0322 -   473   492
=!<>     Abs  921727 #E107F -  2118  2141
=!>R     Abs  920082 #E0A12 -  1274  1298
=!ADD    Abs  917896 #E0188 -   238   257
 !ASC    Ext              -    19
=!AT     Abs  918125 #E026D -   368   386
=!BL     Abs  920718 #E0C8E -  1647  1662
=!C!     Abs  920774 #E0CC6 -  1683  1739
=!C,     Abs  921799 #E10C7 -  2161  2192
=!CAT    Abs  920739 #E0CA3 -  1663  1682
 !CLALL  Ext              -    24
=!COLON  Abs  918619 #E045B -   643   669
=!COMMA  Abs  918232 #E02D8 -   444   472
=!CR     Abs  921398 #E0F36 -  1973  1992
=!D+     Abs  920659 #E0C53 -  1620  1646
=!D-     Abs  920633 #E0C39 -  1605  1619
=!D.     Abs  921271 #E0EB7 -  1919  1935
=!D<     Abs  921580 #E0FEC -  2054  2091
 !DEFIN  Ext              -    27
=!DO     Abs  920897 #E0D41 -  1767  1793
=!DOT    Abs  918374 #E0366 -   510   528
=!DOTP   Abs  921046 #E0DD6 -  1835  1870
=!DOTQ   Abs  920938 #E0D6A -  1794  1834
=!DOTS   Abs  919252 #E06D4 -   928   957
 !END$   Ext              -    20
=!EQUAL  Abs  917956 #E01C4 -   276   297
=!F*     Abs  919796 #E08F4 -  1146  1159
=!F+     Abs  919707 #E089B -  1102  1115
=!F-     Abs  919738 #E08BA -  1116  1145
=!F.     Abs  919446 #E0796 -   996  1060
=!F/     Abs  919827 #E0913 -  1160  1173
 !FADJ   Ext              -    23
=!FIVE+  Abs  920341 #E0B15 -  1427  1445
=!FIVE-  Abs  920376 #E0B38 -  1446  1463
=!FP     Abs  919174 #E0686 -   886   908
=!GT     Abs  918061 #E022D -   332   347
=!GT#    Abs  921488 #E0F90 -  2015  2036
=!GTZ    Abs  920549 #E0BE5 -  1558  1579
=!H.     Abs  919355 #E073B -   958   979
=!I      Abs  917811 #E0133 -   190   213
=!IF     Abs  920846 #E0D0E -  1740  1766
 !IMMED  Ext              -    25
=!IP     Abs  919213 #E06AD -   909   927
=!J      Abs  917840 #E0150 -   214   237
=!L      Abs  917740 #E00EC -   145   158
=!LBRAC  Abs  918340 #E0344 -   493   509
 !LEFT$  Ext              -    21
=!LN     Abs  919411 #E0773 -   980   995
=!LT     Abs  917995 #E01EB -   298   331
=!LT#    Abs  921544 #E0FC8 -  2037  2053
=!LTZ    Abs  920516 #E0BC4 -  1536  1557
=!M*     Abs  921661 #E103D -  2092  2117
=!M/     Abs  921175 #E0E57 -  1888  1918
=!MINUS  Abs  917926 #E01A6 -   258   275
```

```
=!MULT   Abs  918507 #E03EB -   567   582
=!N!     Abs  919666 #E0872 -  1082  1101
=!N#     Abs  918403 #E0383 -   529   566
=!N#S    Abs  921437 #E0F5D -  1993  2014
=!N@     Abs  919630 #E084E -  1061  1081
=!OF     Abs  919858 #E0932 -  1174  1251
=!ONE    Abs  918175 #E029F -   400   413
=!ONE+   Abs  920436 #E0B74 -  1482  1498
=!ONE-   Abs  920490 #E0BAA -  1517  1535
=!OR     Abs  920157 #E0A5D -  1321  1340
=!PAREN  Abs  918722 #E04C2 -   686   727
=!PSTOR  Abs  920224 #E0AA0 -  1361  1384
=!QUEST  Abs  918693 #E04A5 -   670   685
=!QUOTE  Abs  918788 #E0504 -   728    17
=!R>     Abs  920042 #E09EA -  1252  1273
=!R@     Abs  920122 #E0A3A -  1299  1320
=!RBRAC  Abs  918580 #E0434 -   611   642
 !RITE$  Ext                -    22
=!S!     Abs  921870 #E110E -  2193  2229
=!S0     Abs  921144 #E0E38 -  1871  1887
=!S<     Abs  922104 #E11F8 -  2299    18
=!S=     Abs  921958 #E1166 -  2230  2298
=!SEMIC  Abs  918536 #E0408 -   583   610
=!STORE  Abs  918088 #E0248 -   348   367
 !STRAR  Ext                -    28
=!T      Abs  917715 #E00D3 -   130   144
=!THREE  Abs  918213 #E02C5 -   428   443
=!TIC    Abs  917765 #E0105 -   159   189
=!TWO    Abs  918194 #E02B2 -   414   427
=!TWO*   Abs  920272 #E0AD0 -  1385  1401
=!TWO+   Abs  920408 #E0B58 -  1464  1481
=!TWO-   Abs  920462 #E0B8E -  1499  1516
=!TWO/   Abs  920298 #E0AEA -  1402  1426
=!U.     Abs  921768 #E10A8 -  2142  2160
=!U<     Abs  920587 #E0C0B -  1580  1604
 !VOCAB  Ext                -    26
=!X      Abs  917574 #E0046 -    50    16
=!XX     Abs  917640 #E0088 -    82    97
=!Y      Abs  917665 #E00A1 -    98   113
=!Z      Abs  917690 #E00BA -   114   129
=!ZEQ    Abs  920191 #E0A7F -  1341  1360
=!ZERO   Abs  918156 #E028C -   387   399
 $<>     Abs  921738 #E108A -  2123  2122
=$>R     Abs  920093 #E0A1D -  1279  1277
 $ABS    Ext                -  1407
=$ADD    Abs  917905 #E0191 -   242   240
=$AT     Abs  918134 #E0276 -   372   370
=$C!     Abs  920785 #E0CD1 -  1688  1686
=$CAT    Abs  920750 #E0CAE -  1668  1666
=$CR     Abs  921409 #E0F41 -  1978  1976
=$D+     Abs  920670 #E0C5E -  1625  1611  1623
=$D-     Abs  920644 #E0C44 -  1610  1608
=$D<     Abs  921591 #E0FF7 -  2059  2058
 $DNGAT  Ext                -  1610
=$DP     Abs  920832 #E0D00 -  1725  1724
```



```
=$EQUAL  Abs  917965 #E01CD -   280   278
=$F*     Abs  919807 #E08FF -  1150  1149
=$F+     Abs  919718 #E08A6 -  1106  1105
=$F-     Abs  919749 #E08C5 -  1120  1119
=$F.     Abs  919477 #E07B5 -  1009  1008
=$F/     Abs  919838 #E091E -  1164  1163
=$FIVE+  Abs  920352 #E0B20 -  1432  1430
=$FIVE-  Abs  920387 #E0B43 -  1451  1449
=$FP     Abs  919185 #E0691 -   890   889
=$GT     Abs  918070 #E0236 -   336   334
=$GTZ    Abs  920560 #E0BF0 -  1563  1561
=$IP     Abs  919224 #E06B8 -   913   912
 $JJ     Abs  917869 #E016D -   222   221
=$L      Abs  917749 #E00F5 -   149   147
 $LN     Abs  919422 #E077E -   984   983
=$LT     Abs  918004 #E01F4 -   302   300
=$LTZ    Abs  920527 #E0BCF -  1541  1539
=$MINUS  Abs  917935 #E01AF -   262   261
=$N!     Abs  919677 #E087D -  1087  1085
=$N@     Abs  919641 #E0859 -  1066  1064
=$ONE+   Abs  920447 #E0B7F -  1487  1485
=$ONE-   Abs  920501 #E0BB5 -  1522  1520
=$OR     Abs  920168 #E0A68 -  1326  1324
=$PSTOR  Abs  920235 #E0AAB -  1366  1364
=$R>     Abs  920053 #E09F5 -  1257  1255
=$R@     Abs  920133 #E0A45 -  1305  1303
=$S<     Abs  922115 #E1203 -  2304  2302
=$S=     Abs  921969 #E1171 -  2235  2233
=$STORE  Abs  918097 #E0251 -   352   350
=$T      Abs  917724 #E00DC -   134   132
=$TWO*   Abs  920283 #E0ADB -  1390  1388
=$TWO+   Abs  920419 #E0B63 -  1469  1467
=$TWO-   Abs  920473 #E0B99 -  1504  1502
=$TWO/   Abs  920309 #E0AF5 -  1407  1405
 $TYPE   Ext                -  1010
=$U<     Abs  920598 #E0C16 -  1585  1583
 $XSTR$  Abs  919492 #E07C4 -  1018  1009  1017
=$XX     Abs  917649 #E0091 -    86    84
=$Y      Abs  917674 #E00AA -   102   100
=$Z      Abs  917699 #E00C3 -   118   116
=$ZEQ    Abs  920202 #E0A8A -  1346  1344
 $four+  Abs  920021 #E09D5 -  1237  1236
=*/      Abs  921313 #E0EE1 -  1940
 */MOD   Ext                -  1941
 -FIND   Ext                -   162
=/       Abs  918310 #E0326 -   475
 /MOD    Ext                -   477
 2DROP   Ext                -  1198  2020
 2DUP    Ext                -  1999  2097  2197  2211
 <+LP>   Ext                -   947
=<>      Abs  921733 #E1085 -  2122  1205
 <>0     Abs  921758 #E109E -  2131  2128
=>R      Abs  920088 #E0A18 -  1277   864  1894  1895  1962  2099
 ?COMP   Ext                -  1191  1799
 ?NEG    Ext                -  1903  1907
```

```
 ABORTU   Ext                  -   167  1206  2208
 ABRT"X   Ext                  -    59   702   764   935  1817  1856
 ABS      Ext                  -  1898  2100  2102
 AD15M    Ext                  -  1107
=ADD      Abs  917900 #E018C   -   240   549   556   863  1961
 ALLOT    Ext                  -   462   741  2180
=AT       Abs  918129 #E0271   -   370    54    58   449   534   651   674   733
                               -   770   942   963  1876  2022  2168
 AVMEMS   Ext                  -  1026
 BASE     Ext                  -   533   962   967
=BL       Abs  920724 #E0C94   -  1650
 BLK      Ext                  -    53
 BRNCH    Ext                  -   755  1199
=C!       Abs  920780 #E0CCC   -  1686   753   777   827   828  2178  2213
=C,       Abs  921805 #E10CD   -  2165
=CAT      Abs  920745 #E0CA9   -  1666   774   794   809   837   855  1801  1841
 CLRFRC   Ext                  -   916
 CMOV>    Ext                  -   808
 CMOVE    Ext                  -  2217
 COLLAP   Ext                  -  1019
=COLON    Abs  918623 #E045F   -   646
=COMMA    Abs  918236 #E02DC   -   446  1222  1750
 COMPL    Ext                  -   193   217   592   738  1211  1214  1216  1223
                               -  1745  1772  1812
 CONTX    Ext                  -   652
 COUNT    Ext                  -  1860  1956
=CR       Abs  921404 #E0F3C   -  1976   932
 CREAT    Ext                  -   654
 CRLFOF   Ext                  -  1980
 CSP      Ext                  -   648
 CURR     Ext                  -   650
=D+       Abs  920665 #E0C59   -  1623
=D-       Abs  920639 #E0C3F   -  1608
=D.       Abs  921277 #E0EBD   -  1923   515   944  2148
 D.R      Ext                  -  1925
 D0       Abs  920999 #E0DA7   -  1810  1806
 D03      Abs  921036 #E0DCC   -  1822
=D<       Abs  921586 #E0FF2   -  2058
 D<RES    Abs  921643 #E102B   -  2079  2083
 D<TRUE   Abs  921648 #E1030   -  2081  2077
 D?NEG    Ext                  -  2105
 DABS     Ext                  -  1896
 DEPTH    Ext                  -   933
=DO       Abs  920903 #E0D47   -  1771
 DOCOL    Ext                  -    52   161   192   216   446   457   475   496
                               -   513   532   570   586   614   646   660   673
                               -   689   731   847   931   961   999  1190  1744
                               -  1771  1798  1839  1874  1892  1923  1940  1954
                               -  1996  2019  2041  2096  2146  2165  2176  2196
 DOCON    Ext                  -   390   403   417   431  1650
=DOI      Abs  918378 #E036A   -   513   675
=DOTP     Abs  921052 #E0DDC   -  1839
=DOTQ     Abs  920944 #E0D70   -  1798
=DOTS     Abs  919258 #E06DA   -   931
=DOUSE    Abs  920808 #E0CE8   -  1709    86   102   118   134   149  1725
```



```
=DP       Abs  920827 #E0CFB   -  1724
 DROP     Ext                  -    64   169   480   572   706  1224  1943  2200
 DS00     Abs  919305 #E0709   -   941   948
 DUP      Ext                  -   773   788   810   836   852   856  1192  1202
                               -  1957
 DV2-15   Ext                  -  1165
 EQ0      Abs  917985 #E01E1   -   288   286
=EQUAL    Abs  917960 #E01C8   -   278  1195  1215  1804  1844
 EXEC     Ext                  -   450  2169
=F*       Abs  919802 #E08FA   -  1149
=F+       Abs  919713 #E08A1   -  1105
=F-       Abs  919744 #E08C0   -  1119
=F.       Abs  919452 #E079C   -   999
 F.0      Abs  919551 #E07FF   -  1032  1029
=F/       Abs  919833 #E0919   -  1163
 FAIL     Abs  922210 #E1262   -  2345  2248  2332  2340
 FALSE    Abs  918040 #E0218   -   316
=FEND     Abs  919785 #E08E9   -  1135   898   917   986  1108  1152  1166
=FF.      Abs  919472 #E07B0   -  1008  1000
=FIVE+    Abs  920347 #E0B1B   -  1430
=FIVE-    Abs  920382 #E0B3E   -  1449
=FOUR+    Abs  920016 #E09D0   -  1236   800   853
=FP       Abs  919180 #E068C   -   889
 FRAC15   Ext                  -   897
=FSTRT    Abs  919760 #E08D0   -  1127  1106  1120  1150  1164
 FV0      Abs  920364 #E0B2C   -  1435  1454
 GCHAR    Ext                  -   698   759  1814  1853
 GETFP    Ext                  -  1045  1981
 GETXLN   Ext                  -  1132
=GT       Abs  918065 #E0231   -   334
=GT#      Abs  921494 #E0F96   -  2019
 GTLT1    Abs  918013 #E01FD   -   305   339
=GTZ      Abs  920555 #E0BEB   -  1561
 GTZ0     Abs  920577 #E0C01   -  1570  1566  1568
=H.       Abs  919361 #E0741   -   961
 HERE     Ext                  -   458   752   754  1219  1747  1774  2177
 HEX      Ext                  -   965
 HLD      Ext                  -  2021  2043
 HOLD     Ext                  -   557
=I        Abs  920128 #E0A40   -  1302   941
=ICCOMA   Abs  921835 #E10EB   -  2176
=ICOMMA   Abs  918266 #E02FA   -   457
=IF       Abs  920852 #E0D14   -  1744
=II       Abs  917815 #E0137   -   192
 IN       Ext                  -   692
=IP       Abs  919219 #E06B3   -   912
=J        Abs  917844 #E0154   -   216
=JJ       Abs  917864 #E0168   -   221   218
=L        Abs  917744 #E00F0   -   147
=LBRAC    Abs  918344 #E0348   -   496   599
 LESS     Abs  922200 #E1258   -  2342  2249  2330
 LIT      Ext                  -   447   460   537   547   551   615   690   696
                               -   757   819   945  1193  1203  1226  1802  1810
                               -  1842  1851  2166
 LITR     Ext                  -   171
```

```
=LN      Abs  919417 #E0779 -   983
 LN15    Ext                 -   985
=LT      Abs  917999 #E01EF -   300   540  2207
=LT#     Abs  921550 #E0FCE -  2041
=LTZ     Abs  920522 #E0BCA -  1539
 LTZ0    Abs  920539 #E0BDB -  1546  1544
=M*      Abs  921667 #E1043 -  2096
=M/      Abs  921181 #E0E5D -  1892
 M/MOD   Ext                 -   535
 MATCH   Abs  922204 #E125C -  2344  2246
 MAX     Ext                 -   821
 MEMERR  Ext                 -  1030
=MINUS   Abs  917930 #E01AA -   261  2025
 MOVDP   Ext                 -   778  1821
 MP2-15  Ext                 -  1151
 MTHSTK  Ext                 -  1020
=MULT    Abs  918511 #E03EF -   570
 MXLEN   Ext                 -  2198
=N!      Abs  919672 #E0878 -  1085
=N#      Abs  918407 #E0387 -   532  1998
 N#00    Abs  918477 #E03CD -   551   542
=N#S     Abs  921443 #E0F63 -  1996
 N00     Abs  921448 #E0F68 -  1998  2003
=N@      Abs  919636 #E0854 -  1064
 NOWADD  Abs  919722 #E08AA -  1107  1122
 NQOT?   Ext                 -   743
 NUMST   Ext                 -   890   913   984
=OF      Abs  919864 #E0938 -  1190
 OF01    Abs  919919 #E096F -  1202  1197
 OF02    Abs  919946 #E098A -  1211  1200
=ONE     Abs  918179 #E02A3 -   403
=ONE+    Abs  920442 #E0B7A -  1485  1958
=ONE-    Abs  920496 #E0BB0 -  1520
=OR      Abs  920163 #E0A63 -  1324  2000
 OVER    Ext                 -   539   775   790   825  1213  1893  2024
 PAD     Ext                 -   795   823   834  2023  2042
=PAREN   Abs  918726 #E04C6 -   689
 PD0     Abs  921102 #E0E0E -  1851  1846
 PD03    Abs  921134 #E0E2E -  1862
=PDOTQ   Abs  921338 #E0EFA -  1954  1813
 PICK    Ext                 -  2205
=PSTOR   Abs  920230 #E0AA6 -  1364   693
 PT1+    Ext                 -   750  1807  1847
 PUTABX  Ext                 -  1135
 QCSP    Ext                 -   587
=QUEST   Abs  918697 #E04A9 -   673
 QUOT02  Abs  918887 #E0567 -   757   749
 QUOT2   Abs  918837 #E0535 -   743   737
 QUOT3   Abs  918964 #E05B4 -   788   772
 QUOT5   Abs  918909 #E057D -   769   756
=QUOTC   Abs  919104 #E0640 -   847   739
=QUOTE   Abs  918792 #E0508 -   731
=R>      Abs  920048 #E09F0 -  1255    63   862  1900  1906  1960  2104
=R@      Abs  920128 #E0A40 -  1303   194   848  1897  1901  1955
=RBRAC   Abs  918584 #E0438 -   614   658
```



```
=REND1   Abs  922216 #E1268 -  2347
=RESET   Abs  922053 #E11C5 -  2266  2242
 REV$    Ext                 -  1037
 ROM;C   Ext                 -   659
 ROT     Ext                 -   536
=S!      Abs  921876 #E1114 -  2196
 S->D    Ext                 -   514
=S0      Abs  921150 #E0E3E -  1874   938
=S<      Abs  922110 #E11FE -  2302
 S<00    Abs  922164 #E1234 -  2325  2336
 S<01    Abs  922158 #E122E -  2323  2315
 S<02    Abs  922195 #E1253 -  2339  2326
=S=      Abs  921964 #E116C -  2233
=SAME    Abs  922067 #E11D3 -  2276  2243  2285
 SAMENO  Abs  922095 #E11EF -  2286  2281
 SAMEY   Abs  922097 #E11F1 -  2287  2277
 SAMSGN  Abs  918035 #E0213 -   314   311
 SAMSIN  Abs  921636 #E1024 -  2076  2072
 SAVEFP  Ext                 -  1018  1127  1978  2235  2304
 SEMI    Ext                 -    65   175   195   219   451   463   481   500
                                 516   558   573   593   601   619   676   707
                                 781   841   868   949   969  1002  1228  1752
                                1777  1808  1822  1849  1862  1877  1909  1927
                                1944  1964  2004  2027  2045  2106  2149  2170
                                2181  2218
=SEMIC   Abs  918540 #E040C -   586
=SETSTR  Abs  922018 #E11A2 -  2253  2236
 SMUDG   Ext                 -   595   655
 SP0     Ext                 -  1875
 SP@     Ext                 -   647   937
 SPACE   Ext                 -  1001  1926
 STATE   Ext                 -    57   498   617   732   769
 STKDRP  Ext                 -  1130
=STORE   Abs  918092 #E024C -   350   459   499   618   649   653   968  2044
 STR$SB  Ext                 -  1036
 STRM    Ext                 -  1800  1840
 SWAP    Ext                 -   479   801   822   824   838   840   854   939
                                 964  1905  1908  1942  2101  2199  2216
=T       Abs  917719 #E00D7 -   132
 T/02    Abs  920331 #E0B0B -  1417  1415
=TABLE   Abs  917504 #E0000 -    16
=THREE   Abs  918217 #E02C9 -   431  1776  2204
=TIC     Abs  917769 #E0109 -   161
 TRUE    Abs  918047 #E021F -   320   315
=TWO     Abs  918198 #E02B6 -   417   740  1751  2179
=TWO*    Abs  920278 #E0AD6 -  1388   859  1959
=TWO+    Abs  920414 #E0B5E -  1467   789   826   835   839   857
=TWO-    Abs  920468 #E0B94 -  1502   776  2212
=TWO/    Abs  920304 #E0AF0 -  1405  2026
 TYPE    Ext                 -  1861  1963
 U*      Ext                 -   571  2103
=U.      Abs  921774 #E10AE -  2146   966
=U<      Abs  920593 #E0C11 -  1583
 U<TR    Abs  920619 #E0C2B -  1593  1589
 UDIV    Ext                 -  1899
```

```
Saturn Assembler     Jump_Table,_0,1,&2_char_words    Tue Jan 10, 1984   9:16 am
Ver. 3.33/Rev. 2241  Symbol Table                                      Page  91

=X       Abs  917578 #E004A -    52
 X1      Abs  917620 #E0074 -    63    56
 XDO     Ext                -   940  1773
 XOR     Ext                -  1902  2098
=XSTR$   Abs  919487 #E07BF -  1017
=XX      Abs  917644 #E008C -    84
=Y       Abs  917669 #E00A5 -   100
=Z       Abs  917694 #E00BE -   116
 ZBRNH   Ext                -    55   541   736   748   771  1196  1217  1746
                             1805  1845  2002
=ZEQ     Abs  920197 #E0A85 -  1344   166   934  2001
 ZEQ0    Abs  920214 #E0A96 -  1351  1349
=ZERO    Abs  918160 #E0290 -   390   497   751   943  1221  1749  1924  2147
 fFAIL   Ext                -  2345
 fMATCH  Ext                -  2344
 fail    Abs  922010 #E119A -  2248  2240
 less    Abs  922014 #E119E -  2249  2244
 uMODES  Ext                -  1128
```

```
Saturn Assembler     Jump_Table,_0,1,&2_char_words    Tue Jan 10, 1984   9:16 am
Ver. 3.33/Rev. 2241  Statistics                                        Page  92

Input Parameters

  Source file name is MR&FT1

  Listing file name is MR/FT1

  Object file name is MR%FT1::65

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News
```

```
Saturn Assembler     FT2:_3_&_4_CHAR_WORDS            Tue Feb  7, 1984   3:58 pm
Ver. 3.33/Rev. 2241                                                    Page   1

     1                TITLE  FT2:_3_&_4_CHAR_WORDS
     2                RDSYMB MR%GTO
     3          *     RDSYMB MR%FT1
     4          *     ABS    REND1
     5
     6 E1300          ABS    #E1300
     7
     8          ***********************************
     9          *                                 *
    10          *     MR&FT2       <840608.1403> *
    11          *                                 *
    12          * Re-released: 2/6/84             *
    13          * to correct STD                  *
    14          *                                 *
    15          ***********************************
    16
    17          ****************************************
    18          *                                      *
    19          *   START OF 3 CHAR WORDS              *
    20          * ADD ADDITIONAL WORDS AT THIS END     *
    21          *                                      *
    22          ****************************************
```

```
                EJECT
               ****************************************
               *
               * S<&           ( STR1 STR2 -- STR3 )
               *
               * STRING CONCATENATE
               * String2 is appended directly onto the end of
               * string1 (assuming string1's maxlen will permit
               * it). Str3 is the addr of str1, and the new
               * combined length.
               * When total length is too big to fit, give error.
               ****************************************
E1300 0000     CON(5)  0
      0
E1305 38   =!S<&  CON(2)  #83
E1307 35C3        NIBASC  \S<\
E130B 6A          CON(2)  \&\+#80
E130D 0000 =S<&   CON(5)  =DOCOL          STACK:
      0
E1312 B591        CON(5)  =DUP            AD1 L1 AD2 L2 L2
      E
E1317 45C1        CON(5)  =LIT            AD1 L1 AD2 L2 L2
      E
E131C 4000        CON(5)  4               AD1 L1 AD2 L2 L2 4
      0
E1321 1052        CON(5)  =PICK           AD1 L1 AD2 L2 L2 L1
      E
E1326 0000        CON(5)  =ADD            AD1 L1 AD2 L2 LEN3
      0
E132B 45C1        CON(5)  =LIT            AD1 L1 AD2 L2 LEN3
      E
E1330 5000        CON(5)  5               AD1 L1 AD2 L2 LEN3
      0
E1335 1052        CON(5)  =PICK           AD1 L1 AD2 L2 LEN3 AD1
      E
E133A B591        CON(5)  =DUP            AD1 L1 AD2 L2 LEN3 AD1 AD1
      E
E133F 0000        CON(5)  =>R             AD1 L1 AD2 L2 LEN3 AD1
      0
E1344 45C1        CON(5)  =LIT            AD1 L1 AD2 L2 LEN3 AD1
      E
E1349 4000        CON(5)  4               AD1 L1 AD2 L2 LEN3 AD1
      0
E134E 0000        CON(5)  =MINUS          AD1 L1 AD2 L2 LEN3 AD1-4
      0
E1353 0000        CON(5)  =CAT            AD1 L1 AD2 L2 LEN3 MAX1
      0
E1358 3352        CON(5)  =OVER           AD1 L1 AD2 L2 L3 MAX1 L3
      E
E135D 0000        CON(5)  =LT             AD1 L1 AD2 L2 L3 (MAX1<L3)
      0
E1362 0000        CON(5)  =ABORTW
      0
E1367 53          CON(2)  =eSWF           "STRING WON'T FIT"
          *
```



```
E1369 0000        CON(5)  =>R             AD1 L1 AD2 L2:: L3 AD1
      0
E136E 0000        CON(5)  =>R             AD1 L1 AD2 :: L2 L3 AD1
      0
E1373 0000        CON(5)  =>R             AD1 L1 :: AD2 L2 L3 AD1
      0
E1378 0000        CON(5)  =TWO*           CHARS-->NIBS
      0
E137D 0000        CON(5)  =ADD            DEST :: AD2 L2 L3 AD1
      0
E1382 0000        CON(5)  =R>             DEST AD2 :: L2 L3 AD1
      0
E1387 9D42        CON(5)  =SWAP           AD2 DEST :: L2 L3 AD1
      E
E138C 0000        CON(5)  =R>             AD2 DEST L2 :: L3 AD1
      0
E1391 0000        CON(5)  =CMOVE          STRING MOVED
      0
E1396 0000        CON(5)  =R>             L3
      0
E139B 0000        CON(5)  =R>             L3 AD1
      0
E13A0 5552        CON(5)  =2DUP           L3 AD1 L3 AD1
      E
E13A5 0000        CON(5)  =TWO-           L3 AD1 L3 AD1-2
      0
E13AA 0000        CON(5)  =C!             L3 AD1 (NEW LEN STORED)
      0
E13AF 9D42        CON(5)  =SWAP           AD1 L3
      E
E13B4 0000        CON(5)  =SEMI
      0

               ****************************************
               *
               * S>&           ( STR1 STR2 -- STR3 )
               *
               * STRING CONCATENATE
               * Much like S<&, but the resulting string is
               * at the addr of str2. Str2 is first shifted
               * up and str1 is then copied into beginning
               * of str1's space.
               * When str1 won't all fit in, GIVE ERROR.
               ****************************************
E13B9 5031        CON(5)  =!S<&
      E
E13BE 38   =!S>&  CON(2)  #83
E13C0 35E3        NIBASC  \S>\
E13C4 6A          CON(2)  \&\+#80
E13C6 0000 =S>&   CON(5)  =DOCOL
      0
E13CB B591        CON(5)  DUP             AD1 L1 AD2 L2 L2
      E
E13D0 45C1        CON(5)  =LIT            AD1 L1 AD2 L2 L2
      E
```

```
  94 E13D5 4000          CON(5)  4               AD1 L1 AD2 L2 L2 4
           0
  95 E13DA 1052          CON(5)  =PICK           AD1 L1 AD2 L2 L2 L1
           E
  96 E13DF 0000          CON(5)  =ADD            AD1 L1 AD2 L2 L3
           0
  97 E13E4 B591          CON(5)  =DUP            AD1 L1 AD2 L2 L3 L3
           E
  98 E13E9 0000          CON(5)  =>R             AD1 L1 AD2 L2 L3  :: L3
           0
  99 E13EE 45C1          CON(5)  =LIT            AD1 L1 AD2 L2 L3
           E
 100 E13F3 3000          CON(5)  3               AD1 L1 AD2 L2 L3 3
           0
 101 E13F8 1052          CON(5)  =PICK           AD1 L1 AD2 L2 L3 AD2
           E
 102 E13FD 0000          CON(5)  =TWO-           AD1 L1 AD2 L2 L3 AD2-2
           0
 103 E1402 0000          CON(5)  =TWO-           AD1 L1 AD2 L2 L3 AD2-4
           0
 104 E1407 0000          CON(5)  =CAT            AD1 L1 AD2 L2 L3 MAX2
           0
 105 E140C 0000          CON(5)  =GT             AD1 L1 AD2 L2 (L3>MAX2)
           0
 106 E1411 0000          CON(5)  =ABORTW         IF (L3>MAX2)
           0
 107 E1416 53            CON(2)  =eSWF           "STRING WON'T FIT"
 108             *
 109 E1418 8791          CON(5)  =ROT            AD1 AD2 L2 L1 :: L3
           E
 110 E141D 8791          CON(5)  =ROT            AD1 L2 L1 AD2 :: L3
           E
 111 E1422 B591          CON(5)  =DUP            AD1 L2 L1 AD2 AD2 :: L3
           E
 112 E1427 0000          CON(5)  =>R             AD1 L2 L1 AD2 :: AD2 L3
           0
 113 E142C B591          CON(5)  =DUP            AD1 L2 L1 AD2 AD2 :: AD2 L3
           E
 114 E1431 8791          CON(5)  =ROT            AD1 L2 AD2 AD2 L1 :: AD2 L3
           E
 115 E1436 B591          CON(5)  =DUP            AD1 L2 AD2 AD2 L1 L1 :: AD2 L3
           E
 116 E143B 0000          CON(5)  =>R             AD1 L2 AD2 AD2 L1 :: L1 AD2 L3
           0
 117 E1440 0000          CON(5)  =TWO*           AD1 L2 AD2 AD2 L1*2 :: L1 AD2 L3
           0
 118 E1445 0000          CON(5)  =ADD            AD1 L2 AD2 DEST :: L1 AD2 L3
           0
 119 E144A 8791          CON(5)  =ROT            AD1 AD2 DEST L2 :: L1 AD2 L3
           E
 120 E144F 0000          CON(5)  =CMOV>          AD1  ::  L1 AD2 L3
           0
 121 E1454 0000          CON(5)  =R>             AD1 L1 :: AD2 L3
           0
 122 E1459 0000          CON(5)  =R@             AD1 L1 AD2 :: AD2 L3
```



```
           0
 123 E145E 9D42          CON(5)  =SWAP           AD1 AD2 L1 :: AD2 L3
           E
 124 E1463 0000          CON(5)  =CMOVE                  :: AD2 L3
           0
 125 E1468 0000          CON(5)  =R>             AD2 ::  L3
           0
 126 E146D 0000          CON(5)  =R>             AD2 L3
           0
 127 E1472 5552          CON(5)  =2DUP           AD2 L3 AD2 L3
           E
 128 E1477 9D42          CON(5)  =SWAP           AD2 L3 L3 AD2
           E
 129 E147C 0000          CON(5)  =TWO-           AD2 L3 L3 AD2-2
           0
 130 E1481 0000          CON(5)  =C!             AD2 L3 (NEW LEN STORED)
           0
 131 E1486 0000          CON(5)  =SEMI           AD2 L3
           0
 132
```

```
                        EJECT
                *****************************************
                *
                * 4N@   (addr --- n)
                * Like @ and C@, but works on 4 nibbles.
                *
                *****************************************
E148B EB31E             CON(5)  =!S>&
E1490 38        =!4N@   CON(2)  #83
E1492 43E4              NIBASC  \4N\
E1496 0C                CON(2)  \@\+#80

E1498 00000     =4N@    CON(5)  =DOCOL          :
E149D 00000             CON(5)  =AT             @
E14A2 45C1E             CON(5)  =LIT
E14A7 FFFF0             CON(5)  #FFFF           FFFF
E14AC 7A91E             CON(5)  =AND            AND
E14B1 00000             CON(5)  =SEMI           ;
```



```
                        EJECT
                *****************************************
                *
                * HEX   This word sets the current base
                *        to base 16
                *
                *****************************************
E14B6 0941E             CON(5)  !4N@
E14BB 38        =!HEX   CON(2)  #83
E14BD 8454              NIBASC  \HE\
E14C1 8D                CON(2)  \X\+#80
E14C3 00000     =HEX    CON(5)  =DOCOL          :
E14C8 45C1E             CON(5)  =LIT
E14CD 01000             CON(5)  16              16
E14D2 B562E             CON(5)  =BASE           BASE
E14D7 00000             CON(5)  =STORE          !
E14DC 00000             CON(5)  =SEMI           ;
```

```
                        EJECT
                *****************************************
                *
                * LOG  This word takes the log base 10
                *      of the value in the floating point
                *      X register
                *
                *****************************************
E14E1 BB41E     =!HEX   CON(5)  !HEX
E14E6 38        =!LOG   CON(2)  #83
E14E8 C474              NIBASC  \LG\
E14EC 4D                CON(2)  \T\+#80
E14EE 3F41E     =LOG    CON(5)  $LOG
E14F3 8EF120    $LOG    GOSUBL  =NUMST      (see routine in this assembly)
E14F9 8F00000           GOSBVL  =LGT15
E1500 8C0000            GOLONG  =FEND
```



```
                        EJECT
                *****************************************
                *
                * CHS  - This word changes the sign of the
                *        value in the floating point X register.
                *        No stackdrop or change to LASTX.
                *
                *****************************************
E1506 6E41E     =!CHS   CON(5)  !LOG          NO MORE 3 CHAR WORDS
E150B 38        =!CHS   CON(2)  #83
E150D 3484              NIBASC  \CH\
E1511 3D                CON(2)  \S\+#80
E1513 8151E     =CHS    CON(5)  $CHS
E1518 34FDBF2   $CHS    LC(5)   (=oX)+15
E151F 136               CD0EX                   SAVE D0
E1522 1524              A=DAT0  S               GET X SIGN
E1526 05                SETDEC
E1528 BCC               A=-A-1  S               COMPLEMENT SIGN NIBBLE
E152B 04                SETHEX
E152D 1504              DAT0=A  S               PUT CHANGED X SIGN BACK
E1531 136               CD0EX                   RESTORE D0
E1534 03                RTNCC
```

```
206                     EJECT
207                *************************************************
208                *
209                * UM*  This is the new FORTH 83 name for the old U* word.
210                *
211                * U*  (un1 un2 --- ud3) perform unsigned multiplication of
212                *     un1 by un2, leave the double number ud3 on stack
213                *
214                *************************************************
215
216 E1536 B051            CON(5)  =!CHS
       E
217 E153B 38    =!U*      CON(2)  #83             UM*
218 E153D 55D4            NIBASC  \UM\
219 E1541 AA              CON(2)  \*\+#80
220 E1543 8451  =U*       CON(5)  =$U*
       E
221
222 E1548 AF0   =$U*      A=0     W
223 E154B 143             A=DAT1  A               A(A)=n1
224 E154E 174             D1=D1+  5
225 E1551 147             C=DAT1  A               C(A)=n2
226 E1554 AF3             D=0     W
227 E1557 D7              D=C     A               D(A)=n2
228 E1559 AF2             C=0     W
229
230 E155C 822   U*1       SB=0                    CLEAR STICKY BIT FOR TEST
231 E155F 81C             ASRB                    TEST LEAST SIG BIT
232 E1562 832             ?SB=0                   WAS IT A ZERO?
233 E1565 50              GOYES   U*0             YES, NO ADD
234 E1567 A7B             C=C+D   W               C=FINAL TOTAL
235 E156A A77   U*0       D=D+D   W               SHIFT LEFT FOR NEXT ADD
236 E156D 97C             ?A#0    W               ANY MORE DIGITS?
237 E1570 CE              GOYES   U*1             YES
238 E1572 145             DAT1=C  A               STORE LSDs
239 E1575 1C9             D1=D1-  10
240 E1578 15D9            DAT1=C  10              STORE MSDs
241 E157C 174             D1=D1+  5
242 E157F 03              RTNCC
```



```
243                     EJECT
244                *****************************************
245                *
246                * [']  (BRACKET TIC)  ( --- ) Used as:
247                *
248                *       ['] <name>
249                *
250                *       Used in colon definitions to
251                *       compile the cfa of <name> as a
252                *       literal.
253                *
254                *       IMMEDIATE
255                *
256                *****************************************
257
258 E1581 B351            CON(5) !U*              NO MORE 3 CHAR WORDS
       E
259 E1586 3C    =!BRACT   CON(2) #C3
260 E1588 B572            NIBASC \['\
261 E158C DD              CON(2) \]\+#80
262 E158E 0000  =BRACT    CON(5) =DOCOL           :
       0
263 E1593 0000            CON(5) =?COMP           ?COMP  make sure we're compiling
       0
264 E1598 0000            CON(5) =TIC             compile CFA as literal into dictio
       0
265 E159D 0000            CON(5) =SEMI            ;  execute immediately
       0
```

```
                        EJECT
                *****************************************
                *
                * STO   (addr --- ) Store the floating poit
                *       value contained in X at the addr
                *       specified.
                *
                *****************************************
E15A2 6851      =!STO   CON(5) !BRACT
      E
E15A7 38        =!STO   CON(2) #83
E15A9 3545              NIBASC \ST\
E15AD FC                CON(2) \O\+#80
E15AF 4B51      =STO    CON(5) $STO
      E
E15B4 20        =$STO   P=     0
E15B6 340D              LC(5)  =oX
      BF2
E15BD 136               CDOEX                  DO-->X
E15C0 1527              A=DAT0 W               READ VALUE OF X
E15C4 136               CDOEX
E15C7 147               C=DAT1 A               READ ADDR
E15CA 174               D1=D1+ 5               RETURN NOTHING ON DATA STK
E15CD 136               CDOEX                  DO=ADDR
E15D0 1507              DAT0=A W               WRITE VALUE TO ADDR
E15D4 134               D0=C                   RESTORE I IN DO
E15D7 03                RTNCC
```



```
                        EJECT
                *****************************************
                *
                * RCL   (addr --- ) Recall the floating point
                *       value contained at addr and push it
                *       into X on the floating point stack
                *       after first doing a stack lift.
                *
                *****************************************
E15D9 7A51              CON(5) !STO
      E
E15DE 38        =!RCL   CON(2) #83
E15E0 2534              NIBASC \RC\
E15E4 CC                CON(2) \L\+#80
E15E6 BE51      =RCL    CON(5) $RCL
      E
E15EB 8E00      =$RCL   GOSUBL =SAVEFP         SAVE FORTH POINTERS
      00
E15F1 147               C=DAT1 A               GET ADDR
E15F4 136               CDOEX
E15F7 1527              A=DAT0 W               RECALL VALUE
E15FB 100               R0=A
E15FE 8E00              GOSUBL =STKLFT         DO STACK LIFT
      00
E1604 110               A=R0                   RECALL VALUE
E1607 1B0D              D0=(5)  =oX
      BF2
E160E 1507              DAT0=A W               STORE VALUE AT X
E1612 8E00              GOSUBL =GETFP
      00
E1618 174               D1=D1+  5              THROW AWAY ADDR
E161B 03                RTNCC
```

```
                        EJECT
                *****************************************
                *                                       *
                * STD           SET STANDARD DISPLAY FORMAT *
                *                                       *
                *****************************************
E161D ED51E     STD     CON(5)  !RCL
E1622 38        =!STD   CON(2)  #83
E1624 3545      NIBASC  \ST\
E1628 4C        CON(2)  \D\+#80
E162A F261E     =STD    CON(5)  =$STD
E162F 20        =$STD   P=      0
E1631 30C       LCHEX   C       STD CODE FOR SETFMT
E1634 6C10      GOTO    SETGET  GOSBVL =SETFMT; RTNCC

                ***************************************
                *
                * FIX           DISPLAY FORMAT
                *               ( n -- )
                *       SET # OF DECIMAL PLACES
                *
                ***************************************
E1638 2261E     CON(5)  =!STD
E163D 38        =!FIX   CON(2)  #83
E163F 6494      NIBASC  \FI\
E1643 8D        CON(2)  \X\+#80
E1645 A461E     =FIX    CON(5)  =$FIX
E164A 7210      =$FIX   GOSUB   SETDGT
E164E 30D       LCHEX   D
                *       GOTO    SETGET  DROP THROUGH

E1651 8F000     SETGET  GOSBVL  =SETFMT
E1658 03        RTNCC
                *
                * 2/6/84 fix to STD; need these nibbles
                * to ensure that nothing moves!
                *
E165A           BSS     6
                *****************************************
                * SET # OF DIGITS IN FORMAT
                * MAX IS 11; IF PARAMETER IS GREATER, USE 11
                *****************************************
E1660 20        SETDGT  P=      0       part of 2/6/84 fix to STD
E1662 AD0       A=0     M       clear out any possible garbage
                *               for bit shift
E1665 143       A=DAT1  A       GET PARAMETER
E1668 174       D1=D1+  5
E166B 34B0      LC(5)   11
000
E1672 C4        A=A+A   A       CHECK FOR NEGATIVES
```



```
E1674 550       GONC    DGT1
E1677 AF0       A=0     W       NEGATIVE PARAMETER-->0
E167A 81C DGT1  ASRB            RESTORE N
E167D 8BA       ?A<=C   A       # <=11 ?
E1680 40        GOYES   DGT01
E1682 DA        A=C     A       SET TO 11
E1684 34000 DGT01 LC(5) =DSPDGT
E168B 137       CD1EX
E168E 1510      DAT1=A  P       SET SYSTEM REGISTER
E1692 135       D1=C            RESTORE D1
E1695 01        RTN
                *
                * 2/6/84 fix to STD, need this nibble to ensure
                * that nothing moves!
                *
E1697           BSS     1

                ***************************************
                *
                * SCI           SET SCIENTIFIC FORMAT
                *               ( n -- )
                *
                ***************************************
E1698 D361E     CON(5)  =!FIX
E169D 38        =!SCI   CON(2)  #83
E169F 3534      NIBASC  \SC\
E16A3 9C        CON(2)  \I\+#80
E16A5 AA61E     =SCI    CON(5)  =$SCI
E16AA 72BF      =$SCI   GOSUB   SETDGT
E16AE 30E       LCHEX   E
E16B1 6F9F      GOTO    SETGET  GOSBVL =SETFMT; GETFP; RTNCC

                ***************************************
                *
                * ENG           SET ENGINEER FORMAT
                *               ( n -- )
                *
                ***************************************
E16B5 D961E     CON(5)  =!SCI
E16BA 38        =!ENG   CON(2)  #83
E16BC 54E4      NIBASC  \EN\
E16C0 7C        CON(2)  \G\+#80
E16C2 7C61E     =ENG    CON(5)  =$ENG
E16C7 759F      =$ENG   GOSUB   SETDGT
E16CB 30F       LCHEX   F
E16CE 628F      GOTO    SETGET  GOSBVL =SETFMT; GETFP; RTNCC
```

```
                        EJECT
                *************************************************
                *
                * SIN          FLOATING POINT WORD -- SINE
                *
                *************************************************
E16D2 AB61      E       CON(5)  =!ENG
E16D7 38        =!SIN   CON(2)  #83
E16D9 3594              NIBASC  \SI\
E16DD EC                CON(2)  \N\+#80
E16DF 4E61      =SIN    CON(5)  =$SIN
E16E4 7030      =$SIN   GOSUB   NUMST       SAVEFP;GETX+L;UMODES
E16E8 8F00      000     GOSBVL  =SIN15
E16EF 8C00      00      GOLONG  =FEND       PUTABX;GETFP;RTNCC

                *********************************************
                *
                * COS
                *
                *********************************************
E16F5 7D61      E       CON(5)  =!SIN
E16FA 38        =!COS   CON(2)  #83
E16FC 34F4              NIBASC  \CO\
E1700 3D                CON(2)  \S\+#80
E1702 7071      =COS    CON(5)  =$COS
E1707 7D00      =$COS   GOSUB   NUMST
E170B 8F00      000     GOSBVL  =COS15
E1712 8C00      00      GOLONG  =FEND

                *********************************************
                * SUBROUTINE OF MATH WORDS
                *********************************************
E1718 8F00      =NUMST  GOSBVL  =uMODES     set various modes like ANGLE/RADIAN
      000
E171F 8E00      00      GOSUBL  =SAVEFP     save FORTH pointers
E1725 8C00      00      GOLONG  =GETX+L     X IN (A,B); LASTX:=X

                *********************************************
                *
                * TAN   GENT
                *
                *********************************************
E172B AF61      E       CON(5)  =!COS
E1730 38        =!TAN   CON(2)  #83
```



```
E1732 4514              NIBASC  \TA\
E1736 EC                CON(2)  \N\+#80
E1738 D371      =TAN    CON(5)  =$TAN
      E
E173D 77DF      =$TAN   GOSUB   NUMST
E1741 8F00      000     GOSBVL  =TAN15
E1748 8C00      00      GOLONG  =FEND

                *********************************************
                *
                * X^2           X SQUARED
                *
                *********************************************
E174E 0371      E       CON(5)  =!TAN
E1753 38        =!X^2   CON(2)  #83
E1755 85E5              NIBASC  \X^\
E1759 2B                CON(2)  \2\+#80
E175B 0671      =X^2    CON(5)  =$X^2
E1760 74BF      =$X^2   GOSUB   NUMST       SAVEFP;GETX+L
E1764 AF9               C=B     W
E1767 AF7               D=C     W
E176A AF6               C=A     W           COPY (A,B) TO (C,D)
E176D 8F00      000     GOSBVL  =MP2-15     X*X
E1774 8C00      00      GOLONG  =FEND

                *************************************
                *
                * 1/X           RECIPROCAL
                *
                *************************************
E177A 3571      E       CON(5)  =!X^2
E177F 38        =!1/X   CON(2)  #83
E1781 13F2              NIBASC  \1/\
E1785 8D                CON(2)  \X\+#80
E1787 C871      =1/X    CON(5)  =$1/X
E178C 788F      =$1/X   GOSUB   NUMST
E1790 8F00      000     GOSBVL  =1/X15
E1797 8C00      00      GOLONG  =FEND

                *************************************
                *
                * Y^X           Y TO THE XTH POWER
                *
                *************************************
E179D F771              CON(5)  =!1/X
```

```
          E
501 E17A2 38   =!Y^X   CON(2)  #83
502 E17A4 95E5         NIBASC  \Y^\
503 E17A8 8D           CON(2)  \X\+#80
504 E17AA FA71 =Y^X    CON(5)  =$Y^X
          E
505 E17AF 8E00 =$Y^X   GOSUBL  =FSTRT      SAVEFP;STKDRP;GETXLN
          00
506 E17B5 8F00         GOSBVL  =YX2-15
          000
507 E17BC 8C00         GOLONG  =FEND
          00
508
509            ****************************************
510            *
511            * e^X
512            *
513            ****************************************
514 E17C2 2A71         CON(5)  =!Y^X
          E
515 E17C7 38   =!e^X   CON(2)  #83
516 E17C9 54E5         NIBASC  \E^\
517 E17CD 8D           CON(2)  \X\+#80
518 E17CF 4D71 =e^X    CON(5)  =$e^X
          E
519 E17D4 704F =$e^X   GOSUB   NUMST
520 E17D8 8F00         GOSBVL  =EXP15
          000
521 E17DF 8C00         GOLONG  =FEND
          00
522
523            ****************************************
524            *
525            * RUP           ROLL UP 41C STACK
526            *
527            ****************************************
528 E17E5 7C71         CON(5)  =!e^X
          E
529 E17EA 38   =!RUP   CON(2)  #83
530 E17EC 2555         NIBASC  \RU\
531 E17F0 0D           CON(2)  \P\+#80
532 E17F2 7F71 =RUP    CON(5)  =$RUP
          E
533 E17F7 8E00 =$RUP   GOSUBL  =SAVEFP
          00
534 E17FD 1B00         D0=(5)  (=oT)
          CF2
535 E1804 1567         C=DAT0  W
536 E1808 10B          R3=C                SAVE T (R3 SAFE FROM NMOVE)
537 E180B 8E00         GOSUBL  =STKLFT     LIFT STACK
          00
538 E1811 1B0D         D0=(5)  (=oX)
          BF2
539 E1818 11B          C=R3
540 E181B 1547         DAT0=C  W           PUT OLD T IN X
```



```
541 E181F 8E00         GOSUBL  =GETFP
          00
542 E1825 03           RTNCC
543
544            ****************************************
545            *
546            * RDN           ROLL DOWN 41C STACK
547            *
548            ****************************************
549 E1827 AE71         CON(5)  =!RUP
          E
550 E182C 38   =!RDN   CON(2)  #83
551 E182E 2544         NIBASC  \RD\
552 E1832 EC           CON(2)  \N\+#80
553 E1834 9381 =RDN    CON(5)  =$RDN
          E
554 E1839 8E00 =$RDN   GOSUBL  =SAVEFP
          00
555 E183F 1B0C         D0=(5)  (=oLASTX)
          BF2
556 E1846 1567         C=DAT0  W           GET LASTX
557 E184A 10B          R3=C                SAVE IT
558 E184D 8E00         GOSUBL  =STKDRP     DROP STACK
          00
559            * STKDRP ALSO PUTS X INTO LASTX
560 E1853 1B0C         D0=(5)  (=oLASTX)
          BF2
561 E185A 1527         A=DAT0  W           GET OLD X, NOW IN LASTX
562 E185E 11B          C=R3
563 E1861 1547         DAT0=C  W           REPLACE OLD LASTX
564 E1865 1B00         D0=(5)  (=oT)
          CF2
565 E186C 1507         DAT0=A  W               PUT OLD X IN T
566 E1870 8E00         GOSUBL  =GETFP
          00
567 E1876 03           RTNCC
568
569            ****************************************************
570            *
571            * SP@  ( -- addr)  return the addr of the top of the
572            *                  stack just before SP@ was executed
573            *
574            ****************************************************
575 E1878 C281         CON(5)  =!RDN
          E
576 E187D 38   =!SP@   CON(2)  #83             SP@
577 E187F 3505         NIBASC  \SP\
578 E1883 0C           CON(2)  \@\+#80
579 E1885 A881 =SP@    CON(5)  =$SP@
          E
580 E188A 137  =$SP@   CD1EX                   DATA STACK ADDR TO C
581 E188D 135          D1=C
582 E1890 1C4          D1=D1-  5               PUSH VALUE TO DATA
583 E1893 145          DAT1=C  A               STACK
584 E1896 03           RTNCC
```

```
                        EJECT
                *************************************
                *
                * C@+           ( STR -- STR+ CH )
                * GET CHARACTER AND INCREMENT ADDRESS
                * DECREMENT CHAR COUNT, ON NULL STRING
                * SIMPLY RETURN A 0
                *
                *************************************
E1898 D781E     =!SP@   CON(5)  =!SP@
E189D 38        =!C@+   CON(2)  #83
E189F 3404      NIBASC  \C@\
E18A3 BA        CON(2)  \+\+#80
E18A5 0000      =C@+    CON(5)  =DOCOL
E18AA B591E     CON(5)  =DUP            IF NULL JUST RETURN 0
E18AF 0000      CON(5)  =ZBRNH          IF
E18B4 D200      CON(5)  (C@0)-*
E18B9 3352E     CON(5)  =OVER           (addr cnt addr)
E18BE 8791E     CON(5)  =ROT            (cnt addr addr)
E18C3 0000      CON(5)  =TWO+           (cnt addr addr+2)
E18C8 8791E     CON(5)  =ROT            (addr addr+2 cnt)
E18CD 0000      CON(5)  =ONE-           (addr addr+2 cnt-1)
E18D2 8791E     CON(5)  =ROT            (addr+2 cnt-1 addr)
E18D7 0000      CON(5)  =CAT            (addr+2 cnt-1 char)
E18DC 0000      CON(5)  =SEMI           ;   cheat!
E18E1 0000 C@0  CON(5)  =ZERO           0
E18E6 0000      CON(5)  =SEMI
```



```
                        EJECT
                *************************************************
                *
                * SP!  (  --  )  initialize the data stack pointer
                *                from oS0
                *
                *************************************************
E18EB D981E     CON(5)  =!C@+
E18F0 38        =!SP!   CON(2)  #83
E18F2 3505      NIBASC  \SP\
E18F6 1A        CON(2)  \!\+#80
E18F8 DF81E     =SP!    CON(5)  =$SP!
E18FD 1F11BF2   =$SP!   D1=(5)  (=oS0)          PTR ADDR TO DATA STACK
E1904 147       C=DAT1  A               C=TOP OF STACK
E1907 135       D1=C                    STACK INITIALIZED
E190A 03        RTNCC
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER

```
                        EJECT
                *******************************************************
                *
                * RP@  (  -- addr)  leaves current addr of return stack
                *                   pointer on the data stack
                *
                *******************************************************

E190C 0F81E             CON(5)  =!SP!
E1911 38      =!RP@     CON(2)  #83             RP@
E1913 2505              NIBASC  \RP\
E1917 0C                CON(2)  \@\+#80
E1919 E191E   =RP@      CON(5)  =$RP@
E191E D9      =$RP@     C=B     A               COPY RTN STK TO C
E1920 1C4               D1=D1-  5
E1923 145               DAT1=C  A               PUSH VALUE OF RTN STK
E1926 03                RTNCC                   TO DATA STACK

                *************************************************************
                *
                * RP!  (  --  )  initialize the return stack pointer from oR0
                *
                *************************************************************

E1928 1191E             CON(5)  =!RP@
E192D 38      =!RP!     CON(2)  #83
E192F 2505              NIBASC  \RP\
E1933 1A                CON(2)  \!\+#80

E1935 A391E   =RP!      CON(5)  =@RP!
E193A 137     =@RP!     CD1EX                   SAVE DATA STK PTR IN C
E193D 1F61BF2           D1=(5)  (=oR0)          LOAD PTR TO TOP OF STACK ADDR
E1944 143               A=DAT1  A               A=TOS (RETURN STK)
E1947 D8                B=A     A               SAVE IN B
E1949 135               D1=C                    RESTORE DATA STK PTR IN D1
E194C 03                RTNCC
```



```
                        EJECT
                *******************************************************
                *
                * DUP  (n -- n n)  duplicate item on top of data stk
                *
                *******************************************************

E194E D291E             CON(5)  =!RP!
E1953 38      =!DUP     CON(2)  #83             DUP
E1955 4455              NIBASC  \DU\
E1959 0D                CON(2)  \P\+#80
E195B 0691E   =DUP      CON(5)  =$DUP

E1960 147     =$DUP     C=DAT1  A               READ TOP ITEM
E1963 1C4               D1=D1-  5
E1966 145               DAT1=C  A               PUSH ON AGAIN
E1969 03                RTNCC

                *************************************************************
                *
                * ROT  (n1 n2 n3 --- n2 n3 n1)  rotate top 3 items bringing
                *                               deepest to top
                *
                *************************************************************

E196B 3591E             CON(5)  =!DUP
E1970 38      =!ROT     CON(2)  #83             ROT
E1972 25F4              NIBASC  \RO\
E1976 4D                CON(2)  \T\+#80
E1978 D791E   =ROT      CON(5)  =$ROT

E197D 147     =$ROT     C=DAT1  A               POP n3
E1980 174               D1=D1+  5
E1983 143               A=DAT1  A               POP n2
E1986 145               DAT1=C  A               PUSH n3
E1989 174               D1=D1+  5
E198C 147               C=DAT1  A               POP n1
E198F 141               DAT1=A  A               PUSH n2
E1992 1C9               D1=D1-  10
E1995 145               DAT1=C  A               PUSH n1
E1998 03                RTNCC
```

```
                        EJECT
                    ************************
                    *
                    * AND  (n1 n2 --- n3)
                    *
                    ************************

E199A 0791          CON(5)  =!ROT
      E
E199F 38    =!AND   CON(2)  #83             AND
E19A1 14E4          NIBASC  \AN\
E19A5 4C            CON(2)  \D\+#80
E19A7 CA91  =AND    CON(5)  =$AND
      E
E19AC 147   =$AND   C=DAT1  A               POP n2
E19AF 174           D1=D1+  5
E19B2 143           A=DAT1  A               POP n1
E19B5 0EF2          C=C&A   A               n1 AND n2
E19B9 145           DAT1=C  A               PUSH RESULT
E19BC 03            RTNCC

                    ********************************************
                    *
                    * XOR  (n1 n2 --- n3)
                    *
                    * NOTE: This rtne uses 1 level of CPU rtn stk
                    *
                    ********************************************

E19BE F991          CON(5)  =!AND
      E
E19C3 38    =!XOR   CON(2)  #83             XOR
E19C5 85F4          NIBASC  \XO\
E19C9 2D            CON(2)  \R\+#80
E19CB 0D91  =XOR    CON(5)  =$XOR
      E

E19D0 143   =$XOR   A=DAT1  A               POP n2
E19D3 174           D1=D1+  5
E19D6 147           C=DAT1  A               POP n1
E19D9 0EFA          C=C!A   A               n1 OR n2
E19DD 06            RSTK=C                  SAVE ON CPU RTN STK
E19DF 147           C=DAT1  A               n1
E19E2 FE            C=-C-1  A               NOT n1
E19E4 FC            A=-A-1  A               NOT n2
E19E6 0EFE          A=A!C   A               NOT n1 OR NOT n2
E19EA 07            C=RSTK
E19EC 0EF2          C=C&A   A               n1 XOR n2
E19F0 145           DAT1=C  A               PUSH ON DATA STK
E19F3 03            RTNCC
```



```
                        EJECT
                    ************************************************
                    *
                    * NOT (n --- <1's compliment of n>
                    *
                    ************************************************

E19F5 3C91          CON(5)  =!XOR
      E
E19FA 38    =!NOT   CON(2)  #83
E19FC E4F4          NIBASC  \NO\
E1A00 4D            CON(2)  \T\+#80
E1A02 70A1  =NOT    CON(5)  =$NOT
      E
E1A07 143   =$NOT   A=DAT1  A               READ VALUE
E1A0A FC            A=-A-1  A               1'S COMPLEMENT
E1A0C 141           DAT1=A  A               write value
E1A0F 03            RTNCC

                    **********************
                    *
                    * ABS (n1 -- |n1|)
                    *
                    **********************

E1A11 AF91          CON(5)  =!NOT
      E
E1A16 38    =!ABS   CON(2)  #83              ABS
E1A18 1424          NIBASC  \AB\
E1A1C 3D            CON(2)  \S\+#80
E1A1E 32A1  =ABS    CON(5)  =$ABS
      E

E1A23 143   =$ABS   A=DAT1  A               # TO A
E1A26 24            P=      4               SET HIGH BIT IN C(A)
E1A28 308           LC(1)   8               FOR SIGN TEST
E1A2B 0E02          C=C&A   P               0 IF POSITIVE
E1A2F 90A           ?C=0    P               # POSITIVE?
E1A32 90            GOYES   AB01            CLEAR CARRY & RETURN
E1A34 20            P=      0               FLAG FOR OTHER ROUTINES
E1A36 F8            A=-A    A               2'S COMP OF NEG #
E1A38 141           DAT1=A  A               STORE ABS VALUE
E1A3B 03    AB01    RTNCC
```

```
793                EJECT
794             ********************************************
795             *
796             * SP0 ( -- addr)  addr of var containing
797             *                 addr of top of data stack
798             *
799             ********************************************
800 E1A3D 61A1          CON(5)  =!ABS
        E
801 E1A42 38    =!SP0   CON(2)  #83             SP0
802 E1A44 3505          NIBASC  \SP\
803 E1A48 0B            CON(2)  \0\+#80
804 E1A4A F4A1 =SP0     CON(5)  =$SP0
        E
805 E1A4F 8E00 =$SP0    GOSUBL  =DOUSE          START OF DATA STACK
        00
806 E1A55 11BF          CON(5)  (=UVAR)
        2
807
808             ********************************************
809             *
810             * RP0  ( --- addr)  returns addr of addr of
811             *      top of return stack
812             *
813             ********************************************
814
815 E1A5A 24A1          CON(5)  =!SP0
        E
816 E1A5F 38    =!RP0   CON(2)  #83             RP0
817 E1A61 2505          NIBASC  \RP\
818 E1A65 0B            CON(2)  \0\+#80
819 E1A67 C6A1 =RP0     CON(5)  =$RP0
        E
820
821 E1A6C 8E00 =$RP0    GOSUBL  =DOUSE          START OF RETURN STACK
        00
822 E1A72 61BF          CON(5)   (=oR0)
        2
823
824             ********************************************
825             *
826             * TIB  ( -- addr)  returns addr of
827             *      the terminal input buffer
828             *
829             ********************************************
830
831 E1A77 F5A1          CON(5)  =!RP0
        E
832 E1A7C 38    =!TIB   CON(2)  #83             TIB
833 E1A7E 4594          NIBASC  \TI\
834 E1A82 2C            CON(2)  \B\+#80
835 E1A84 98A1 =TIB     CON(5)  =$TIB
        E
836
837 E1A89 20    =$TIB   P=      0
```



```
838 E1A8B 34B1          LC(5)   =oTIB
        BF2
839 E1A92 136           CD0EX                   D0-->ADDR OF TIB, C=I
840 E1A95 142           A=DAT0  A               A(A)=TIB ADDR
841 E1A98 1C4           D1=D1-  5
842 E1A9B 141           DAT1=A  A               PUSH TO DATA STACK
843 E1A9E 134           D0=C                    RESTORE I TO D0
844 E1AA1 03            RTNCC
```

```
                        EJECT
                ****************************************
                *
                * USE  ( --- addr ) returns the addr of the
                *       addr of the next data buffer to use
                *
                ****************************************

E1AA3 C7A1E             CON(5)  =!TIB
E1AA8 38        =!USE   CON(2)  #83             USE
E1AAA 5535              NIBASC  \US\
E1AAE 5C                CON(2)  \E\+#80
E1AB0 5BA1E     =USE    CON(5)  =$USE

E1AB5 8E00      =$USE   GOSUBL  =DOUSE
      00
E1ABB 02BF2             CON(5)   (=oUSE)

                ****************************************
                *
                * BLK   ( --- addr) returns the addr of the
                *       variable holding the current line
                *       number being interpreted, or 0
                *
                ****************************************

E1AC0 8AA1E             CON(5)  =!USE
E1AC5 38        =!BLK   CON(2)  #83             BLK
E1AC7 24C4              NIBASC  \BL\
E1ACB BC                CON(2)  \K\+#80
E1ACD 2DA1E     =BLK    CON(5)  =$BLK

E1AD2 8E00      =$BLK   GOSUBL  =DOUSE
      00
E1AD8 25BF2             CON(5)   (=oBLK)

                ****************************************
                *
                * >IN  ( --- addr)  returns the addr of the
                *       variable holding the current pointer
                *       into the line being interpreted
                *
                ****************************************

E1ADD 5CA1E             CON(5)  =!BLK
E1AE2 38        =!IN    CON(2)  #83             >IN
E1AE4 E394              NIBASC  \>I\
E1AE8 EC                CON(2)  \N\+#80
```



```
E1AEA FEA1E     =IN     CON(5)  =$IN

E1AEF 8E00      =$IN    GOSUBL  =DOUSE
      00
E1AF5 75BF2             CON(5)   (=oIN)
```

```
                        EJECT
                *****************************************
                *
                * DPL  ( --- addr) addr of a variable used
                *       in determining the number of digits
                *       to the right of the decimal point in
                *       double integer math
                *
                *****************************************

E1AFA FFA1 =DPL     CON(5)  =$DPL
      E
E1AFF 8E00 =$DPL    GOSUBL  =DOUSE
      00
E1B05 A7BF          CON(5)   (=oDPL)
      2

                *************************************
                *
                * FLD  ( --- addr) addr of a variable
                *       giving field length in formating
                *       a number for display
                *
                *************************************

E1B0A F0B1 =FLD     CON(5)  =$FLD
      E
E1B0F 8E00 =$FLD    GOSUBL  =DOUSE
      00
E1B15 F7BF          CON(5)   (=oFLD)
      2
```



```
                        EJECT
                *************************************
                *
                * CSP ( --- addr)  addr of variable which
                *       contains stack addr at start of
                *       compilation.
                *
                *************************************

E1B1A F1B1 =CSP     CON(5)  =$CSP
      E
E1B1F 8E00 =$CSP    GOSUBL   =DOUSE
      00
E1B25 48BF          CON(5)   (=oCSP)
      2

                *****************************************
                *
                * HLD  ( --- addr)  addr of variable which
                *       holds addr of latest char output
                *       during a numeric display
                *
                *****************************************

E1B2A F2B1 =HLD     CON(5)  =$HLD
      E
E1B2F 8E00 =$HLD    GOSUBL  =DOUSE
      00
E1B35 98BF          CON(5)   (=oHLD)
      2
```

```
                        EJECT
                *************************************************
                *
                * BYE           exit FORTH
                *
                * Restore saved program status
                * Clear the ACTIVE flag
                * If an error has occurred (flag from ABORT if from FORTHX), then
                * set the system no-continue flag
                *
                * Save the current top of data stack in DSTKAD
                * and GO TO MAIN LOOP
                *
                *************************************************

E1B3A 2EA1E             CON(5)  =!IN
E1B3F 38        =!BYE   CON(2)  #83             BYE
E1B41 2495              NIBASC  \BY\
E1B45 5C                CON(2)  \E\+#80
E1B47 C4B1E     =BYE    CON(5)  =$BYE

E1B4C 7410      =$BYE   GOSUB   BYE1
E1B50 1BDFAF2           D0=(5)  =DSTKAD         SAVE TOS
E1B57 137               CD1EX
E1B5A 144               DATA0=C A
E1B5D 8D000             GOVLNG  =MAINLP         goto to BASIC main loop code

                * BYE1 called by FORTHX

E1B64 20        =BYE1   P=      0
E1B66 7B30              GOSUB   =GTSTAT     RESTORE SAVED PROGRAM STATUS
E1B6A 1BC0BF2           D0=(5)  =ACTIVE
E1B71 D2                C=0     A               CLEAR FORTH ACTIVE FLAG
E1B73 144               DATA0=C A
E1B76 1BBBBF2           D0=(5)  =oERR?          DID ERROR HAPPEN?
E1B7D 146               C=DAT0  A
E1B80 8AA               ?C=0    A
E1B83 00                RTNYES                  no so just return
E1B85 85E               ST=1    14              NO CONTINUE (FOR BENEFIT
E1B88 01                RTN                     OF FORTHX "XYZ" IN PROGRAM)

E1B8A 1B38CF2   =GTS13  D0=(5)  =oPSTAT
E1B91 142               A=DAT0  A
E1B94 814               ASRC                ONLY ONE NIBBLE MATTERS
E1B97 84D               ST=0    13
E1B9A A44               A=A+A   S
E1B9D 550               GONC    GT1
E1BA0 85D               ST=1    13
```



```
E1BA3 01        GT1     RTN

                *****************************************************************
                *
                * RESTORE S13&S14 (PROG-RUNNING & NO-CONTINUE) TO THE WAY THEY
                * WERE WHEN BASIC WAS LAST RUNNING; TURN ON SUSP OR PRGM
                * ANNUNCIATORS IF THEY WERE ON AT THAT TIME.
                *
                *****************************************************************

E1BA5 71EF      =GTSTAT GOSUB   GTS13
E1BA9 840               ST=0    =NoCont
E1BAC A44               A=A+A   S
E1BAF 550               GONC    GT2
E1BB2 850               ST=1    =NoCont
E1BB5 A44       GT2     A=A+A   S
E1BB8 100               R0=A
E1BBB 5D0               GONC    GT3
E1BBE 3100              LC(2)   =flPRGM
E1BC2 8F000             GOSBVL  =SFLAGS
E1BC9 110       GT3     A=R0
E1BCC A44               A=A+A   S
E1BCF 501               GONC    GT4
E1BD2 3100              LC(2)   =flSUSP
E1BD6 8F000             GOSBVL  =SFLAGS
E1BDD 850               ST=1    =NoCont     IF PRGM SUSPENDED, DON'T CONTINUE
E1BE0 01        GT4     RTN
```

```
1020                    EJECT
1021                ******************************************************
1022                *
1023                * KEY  ( --- n) return the key code of the key pressed
1024                *       on the data stack
1025                *
1026                ******************************************************
1027 E1BE2 F3B1E           CON(5)  =!BYE
1028 E1BE7 38     =!KEY    CON(2)  #83             KEY
1029 E1BE9 B454            NIBASC  \KE\
1030 E1BED 9D              CON(2)  \Y\+#80
1031 E1BEF 4FB1E   =KEY    CON(5)  =$KEY
1032
1033 E1BF4 8F000   =$KEY   GOSBVL  =SAVEFP         SAVE FORTH POINTERS
1034 E1BFB 8F000           GOSBVL  =KEYRD
1035
1036                *                               ADDR IN DEFADR+3
1037 E1C02 8F000           GOSBVL  =GETFP          RESTORE FORTH POINTERS
1038 E1C09 137             CD1EX
1039 E1C0C 1F000           D1=(5)  (=DEFADR)+3     ADDR OF CHAR ADDR
1040 E1C13 143             A=DAT1  A               READ CHAR ADDR
1041 E1C16 131             D1=A
1042 E1C19 D0              A=0     A
1043 E1C1B 14B             A=DAT1  B               READ CHAR TO A(A)
1044 E1C1E 135             D1=C
1045 E1C21 1C4             D1=D1-  5
1046 E1C24 141             DAT1=A  A               PUSH CHAR
1047 E1C27 03              RTNCC
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER


```
1048                    EJECT
1049                ******************************************************
1050                *
1051                * PAD ( --- addr)  Address of scratch area used to hold
1052                *       char strings for intermediate processing.
1053                *
1054                *  NOTE: we allow space for 45 characters; this will
1055                *        cover the extreme case of a 40 bit number
1056                *        displayed in binary.
1057                *
1058                ******************************************************
1059
1060 E1C29 7EB1E       CON(5)  !KEY
1061 E1C2E 38     =!PAD CON(2)  #83             PAD
1062 E1C30 0514        NIBASC  \PA\
1063 E1C34 4C          CON(2)  \D\+#80
1064
1065 E1C36 00000 =PAD   CON(5)  =DOCOL           :
1066 E1C3B 95B2E       CON(5)   =HERE           HERE
1067 E1C40 45C1E       CON(5)   =LIT            LIT
1068 E1C45 A5000       CON(5)  90               45 CHARS
1069 E1C4A 00000       CON(5)   =ADD            +
1070 E1C4F 00000       CON(5)   =SEMI           ;
1071
1072                ******************************************************
1073                *
1074                * LIT  ( --- n) Pushes contents of next dictionary addr
1075                *       onto the data stack.
1076                *
1077                *        This word is a headerless primitive!
1078                *
1079                ******************************************************
1080
1081 E1C54 95C1E =LIT    CON(5)  =$LIT
1082
1083 E1C59 146  =$LIT   C=DAT0  A               C=VALUE
1084 E1C5C 164          D0=D0+  5               INCREMENT I
1085 E1C5F 1C4          D1=D1-  5
1086 E1C62 145          DAT1=C  A               PUSH VALUE TO DATA STK
1087 E1C65 03           RTNCC
```

```
1088                    EJECT
1089                ********************************************
1090                *
1091                * XDO  (limit index --- )
1092                *
1093                *       Runtime code for DO of DO-LOOP
1094                *
1095                ********************************************
1096 E1C67 C6C1 =XDO    CON(5)  =$XDO
          E
1097 E1C6C D9   =$XDO   C=B     A
1098 E1C6E 136         CD0EX                   RTN STK TO D0
1099 E1C71 D5          B=C     A               SAVE I IN B
1100 E1C73 143         A=DAT1  A               POP INDEX
1101 E1C76 174         D1=D1+  5               PT TO LIMIT
1102 E1C79 147         C=DAT1  A               POP LIMIT
1103 E1C7C 174         D1=D1+  5               RTN NOTHING ON DATA STK
1104 E1C7F 184         D0=D0-  5
1105 E1C82 144         DAT0=C  A               PUSH LIMIT TO RTN STK
1106 E1C85 184         D0=D0-  5
1107 E1C88 140         DAT0=A  A               PUSH INDEX TO RTN STK
1108 E1C8B D9          C=B     A
1109 E1C8D 136         CD0EX                   D0=I, C=RTN STK
1110 E1C90 D5          B=C     A
1111 E1C92 03          RTNCC
```



```
1112                    EJECT
1113                ********************************************************
1114                *
1115                * MOD  (n1 n2 -- n3)  Divide n1 by n2 leave remainder n3
1116                *       with same sign as n1.
1117                *
1118                ********************************************************
1119
1120 E1C94 E2C1          CON(5)  =!PAD
          E
1121 E1C99 38    =!MOD   CON(2)  #83
1122 E1C9B D4F4          NIBASC  \MO\
1123 E1C9F 4C            CON(2)  \D\+#80
1124
1125 E1CA1 0000 =MOD     CON(5)  =DOCOL          :
          0
1126 E1CA6 C7B2          CON(5)   =/MOD         rem quot
          E
1127 E1CAB 0C42          CON(5)   =DROP         rem
          E
1128 E1CB0 0000          CON(5)   =SEMI          ;
          0
1129
```

```
1130                    EJECT
1131                ******************************************************
1132                *
1133                * D.R ( d n - ) Display d converted according to BASE,
1134                * right aligned in an n character field.  Display the
1135                * sign only if negative.
1136                *
1137                ******************************************************
1138 E1CB5 99C1           CON(5)  =!MOD
          E
1139 E1CBA 38     =!D.R   CON(2)  #83             D.R
1140 E1CBC 44E2           NIBASC  \D.\
1141 E1CC0 2D             CON(2)  \R\+#80
1142
1143 E1CC2 0000 =D.R      CON(5) =DOCOL
          0
1144 E1CC7 0000           CON(5)  =DEPTH          if depth is < 3 we have
          0
1145 E1CCC 0000           CON(5)  =THREE          an error condition, give
          0
1146 E1CD1 0000           CON(5)  =LT             empty stack message
          0
1147 E1CD6 0000           CON(5)  =ABRT"X
          0
1148 E1CDB E0             CON(2)  =eEMPT
1149 E1CDD 0000           CON(5)  =>R             push n to rtn stk
          0
1150 E1CE2 1BE2           CON(5)   =STR$          turn d into temp string
          E
1151 E1CE7 0000           CON(5)  =R>             addr len n
          0
1152 E1CEC 3352           CON(5)  =OVER           addr len n len
          E
1153 E1CF1 0000           CON(5)  =MINUS          addr len (diff n-len)
          0
1154 E1CF6 0000           CON(5)  =SPACS          put out spaces if diff is positive
          0
1155 E1CFB 29C2           CON(5)  =TYPE           type rest of string
          E
1156 E1D00 0000           CON(5)  =SEMI
          0
```



```
1157                    EJECT
1158                ********************************
1159                *
1160                * MIN  (n1 n2 --- n3)
1161                *
1162                ********************************
1163
1164 E1D05 ABC1           CON(5)  =!D.R
          E
1165 E1D0A 38     =!MIN   CON(2)  #83
1166 E1D0C D494           NIBASC  \MI\
1167 E1D10 EC             CON(2)  \N\+#80
1168
1169 E1D12 0000 =MIN      CON(5)  =DOCOL          :
          0
1170 E1D17 5552           CON(5)   =2DUP          n1 n2 n1 n2
          E
1171 E1D1C 0000           CON(5)   =GT            n1 n2 flag
          0
1172 E1D21 0000           CON(5)   =ZBRNH         IF n1>n2
          0
1173 E1D26 A000           CON(5)  (MIN1)-*
          0
1174 E1D2B 9D42           CON(5)   =SWAP          n2 n1
          E
1175                *                             THEN
1176 E1D30 0C42 MIN1      CON(5)   =DROP          DROP larger value
          E
1177 E1D35 0000           CON(5)   =SEMI           ;
          0
1178
1179                ****************************
1180                *
1181                * MAX  (n1 n2 --- n3)
1182                *
1183                ****************************
1184
1185 E1D3A A0D1           CON(5)  =!MIN
          E
1186 E1D3F 38     =!MAX   CON(2)  #83
1187 E1D41 D414           NIBASC  \MA\
1188 E1D45 8D             CON(2)  \X\+#80
1189
1190 E1D47 0000 =MAX      CON(5)  =DOCOL          :
          0
1191 E1D4C 5552           CON(5)   =2DUP          n1 n2  n1 n2
          E
1192 E1D51 0000           CON(5)   =LT            n1 n2 flag
          0
1193 E1D56 0000           CON(5)   =ZBRNH         IF n1<n2
          0
1194 E1D5B A000           CON(5)  (MAX1)-*
          0
1195 E1D60 9D42           CON(5)   =SWAP          n2 n1
          E
```

```
1196                *                               THEN
1197 E1D65 0C42 MAX1    CON(5)   =DROP             DROP smaller
           E
1198 E1D6A 0000         CON(5)   =SEMI             ;
           0
```



```
1199                EJECT
1200            ****************************************************************
1201            *
1202            * PFA  (nfa --- pfa)  Convert the name field addr of a compiled
1203            *      definition to its parameter field addr.
1204            *
1205            ****************************************************************
1206
1207 E1D6F 0000 =PFA     CON(5)   =DOCOL            :
           0
1208 E1D74 0000         CON(5)    =ONE             flag to TRAVERSE, move up memory
           0
1209 E1D79 0000         CON(5)    =TRAVS           move to end of name field
           0
1210 E1D7E 45C1         CON(5)    =LIT
           E
1211 E1D83 7000         CON(5)   7                 2 NIBS FOR LAST CHAR, 5 FOR
           0
1212                *                               FOR CFA
1213 E1D88 0000         CON(5)    =ADD             we're pointing at PFA
           0
1214 E1D8D 0000         CON(5)    =SEMI            ;
           0
```

```
1215                    EJECT
1216                ************************************************************
1217                *
1218                * R/W     ( addr  line#  fib#   f --- )
1219                *         This routine reads a line number into a buffer. The
1220                *         file can be in the machine or out somewhere on HPIL
1221                *
1222                *       NO HEADER -- subroutine of BLOCK
1223                *
1224                * NOTE:   the addr is the buffer data addr and not the addr of
1225                *         the start of the buffer.
1226                *
1227                * NOTE:   Currently this routine does not write; it simply
1228                *         returns.
1229                *
1230                ************************************************************
1231
1232 E1D92 79D1 =R/W     CON(5)  =$R/W
       E
1233
1234                * this checking for read/write flag is no longer necessary since
1235                * this routine does not write to a file.  If we ever become press
1236                * for space we can get rid of this check, and get rid of code in
1237                * BUFFER etc. where we push this flag
1238
1239 E1D97 143  =$R/W  A=DAT1  A
1240 E1D9A 174         D1=D1+   5              POP FLAG
1241 E1D9D 8AC         ?A#0     A              IS THIS A WRITE?
1242 E1DA0 60          GOYES    READR          NO
1243 E1DA2 6F60        GOTO     WRITE          YES
1244 E1DA6 8E00 READR  GOSUBL   =SAVEFP        SAVE OUR FORTH POINTERS
       00
1245 E1DAC 133         AD1EX                   COPY DATA STK PTR TO A
1246 E1DAF 130         D0=A                    AND THEN TO D0
1247 E1DB2 AF0         A=0      W
1248 E1DB5 142         A=DAT0   A              A(A)=fib#, D0=DATA STK PTR
1249 E1DB8 164         D0=D0+   5
1250 E1DBB 136         CD0EX                   SAVE D0 IN R0
1251 E1DBE 108         R0=C
1252 E1DC1 8F00        GOSBVL   =FIBAD-        FIND THE fib# START ADDR
       000
1253                *                          SYS WILL RETURN ERROR IF
1254                *                          fib# NONEXISTENT
1255                *                          IT ALSO RESTORE D0 FROM R0
1256                * COMPARE LINE #'S
1257                *
1258 E1DC8 101         R1=A                    SAVE ADDR
1259 E1DCB 1FD9 ANOTH  D1=(5)   =oLINE#        CURRENT LINE# ADDR
       BF2
1260 E1DD2 143         A=DAT1   A              CLINE#
1261 E1DD5 146         C=DAT0   A              DESIRED LINE#
1262 E1DD8 164         D0=D0+   5              POINT TO BUFFER ADDR
1263 E1DDB 8A2         ?A=C     A              SAME NUMBERS?
1264 E1DDE E2          GOYES    ENDR
1265 E1DE0 8B2         ?A<C     A              CLINE# < DESIRED LINE# ?
```



```
1266 E1DE3 60          GOYES    NOTEND         YES, READ A LINE
1267 E1DE5 6B90        GOTO     REPOS          CLINE#>DESIRED LINE, REWIND
1268                *                          FILE & START READING
1269
1270
1271 E1DE9 E4   NOTEND A=A+1    A              UPDATE CLINE#
1272 E1DEB 141         DAT1=A   A              WRITE OUT NEW LINE#
1273 E1DEE 8A2         ?A=C     A              READ THIS ONE?
1274 E1DF1 62          GOYES    READIT         YES
1275 E1DF3 7AA0        GOSUB    REDREC         GET READY FOR FAKE READ
1276 E1DF7 541         GONC     ENDR           EOF, JUMP OUT
1277 E1DFA 490         GOC      NORED0         SET UP FOR READ
1278 E1DFD 8F00 NORED  GOSBVL   =D0+2RD        READ BYTE OF RECORD
       000
1279
1280 E1E04 CD   NORED0 B=B-1    A              DECREMENT CHAR CNT OF REC
1281 E1E06 56F         GONC     NORED          NOT DONE, READ ANOTHER
1282 E1E09 4F4         GOC      TRA0           FINISH UP
1283
1284 E1E0C      ENDR
1285 E1E0C 8E00        GOSUBL   =GETFP         RECOVER FORTH POINTERS
       00
1286 E1E12 17E  WRITE  D1=D1+   15             DROP ITEMS FROM STACK
1287 E1E15 03          RTNCC
1288
1289 E1E17 7680 READIT GOSUB    REDREC         GET READY FOR REC READ
1290 E1E1B 50F         GONC     ENDR           EOF!
1291 E1E1E 3493        LC(5)    =oRSIZE        GET ADDR OF MAX REC LENGTH
       BF2
1292 E1E25 136         CD0EX                   ADDR TO D0
1293 E1E28 14A         A=DAT0   B              A(B)=MAX REC LENGTH ALLOWED
1294 E1E2B 134         D0=C                    RESTORE D0 PTR TO SOURCE
1295 E1E2E 6310        GOTO     RDBYT0         SET UP FOR READ
1296
1297 E1E32 14E  RDBYT  C=DAT0   B              READ BYTE
1298 E1E35 14D         DAT1=C   B              WRITE BYTE
1299 E1E38 171         D1=D1+   2              INCREMENT DEST
1300 E1E3B 8F00        GOSBVL   =D0+2RD        INCREMENT SOURCE
       000
1301
1302 E1E42 CD   RDBYT0 B=B-1    A              DECREMENT CHAR CNT
1303 E1E44 441         GOC      TRA0           DONE-OKAY
1304 E1E47 A6C         A=A-1    B              DECREMENT MAX CHAR CNT
1305 E1E4A 57E         GONC     RDBYT          DO ANOTHER
1306
1307                *
1308                * RECORD IS LONGER THAN ALLOWED: THROW AWAY EXTRA CHARS
1309                *
1310
1311 E1E4D 8F00 TRASH  GOSBVL   =D0+2RD        READ BYE FROM FILE
       000
1312 E1E54 CD          B=B-1    A              DEC CHAR COUNT
1313 E1E56 56F         GONC     TRASH          NOT DONE YET
1314 E1E59 DB   TRA0   C=D      A              TEST FOR ODD REC
1315 E1E5B 8AA         ?C=0     A
```

```
1316 E1E5E 90          GOYES   TRA00
1317 E1E60 8F00        GOSBVL  =D0+2RD          READ GARBAGE NULL
           000
1318 E1E67 D0   TRA00  A=0     A               WRITE 2 NULLS TO
1319 E1E69 1593        DAT1=A  4               END OF RECORD
1320 E1E6D 8F00        GOSBVL  =UPCPOS         UPDATE POSITION IN
           000
1321           *                               FIB FOR THIS FILE
1322
1323 E1E74 112  TRA1   A=R2
1324 E1E77 132         AD0EX                   DATA STK PTR IN D0
1325 E1E7A 184         D0=D0-  5               PT TO LINE#
1326 E1E7D 6D4F        GOTO    ANOTH           SEE IF WE'RE DONE YET
1327
1328 E1E81 132  REPOS  AD0EX
1329 E1E84 102         R2=A
1330 E1E87 111         A=R1                    A=fib#
1331 E1E8A 8F00        GOSBVL  =REWIND         RESET FILE TO BEGINNING
           000
1332 E1E91 1FD9        D1=(5)  =oLINE#
           BF2
1333 E1E98 D0          A=0     A               SET CURRENT LINE # TO 0
1334 E1E9A 141         DAT1=A  A               THIS WILL FORCE A READ
1335 E1E9D 66DF        GOTO    TRA1
1336
1337           *
1338           * This routine sets everything up to start reading a line
1339           * if it encounters EOF it will write set LINE# to the
1340           * desired line #, write 0 length to the byte count field
1341           * of the buffer, write 2 nulls of data to the buffer and
1342           * return with the carry clear.  Otherwise, it returns with
1343           * the carry set and B(A)=actual byte count for the current line.
1344           *
1345
1346 E1EA1 136  REDREC CD0EX                   C=DATA STK, D0-->BUFF
1347 E1EA4 10A         R2=C                    SAVE DATA STK PTR IN R2
1348 E1EA7 8E00        GOSUBL  =GETPTR         SET UP fib POINTERS FOR READ
           00
1349 E1EAD 112         A=R2                    A=DATA STK PTR
1350 E1EB0 132         AD0EX                   DATA STK PTR BACK IN D0
1351 E1EB3 146         C=DAT0  A               C(A)=BUFFER ADDR
1352 E1EB6 137         CD1EX                   D1-> BUFFER SPACE
1353 E1EB9 1C3         D1=D1-  4               D1-->BYTE CNT IN BUFFER
1354
1355           *
1356           * READ A RECORD
1357           *
1358 E1EBC 110         A=R0                    A=ABS POSITION IN FILE
1359 E1EBF 130         D0=A                    D0-->INTO FILE
1360 E1EC2 8F00        GOSBVL  =RDLNAS         READ RECORD LENGTH
           000
1361 E1EC9 23          P=      3
1362 E1ECB B14         A=A+1   WP
1363 E1ECE 523         GONC    RREC5           WILL CARRY IF EOF
1364 E1ED1 20          P=      0               SET P=0 THEN JUMP TO
```



```
1365           *                               BACK2B WHICH WILL
1366           *                               BACK UP 2 BYTES!
1367 E1ED3 8F00        GOSBVL  =BACK2B          BACK UP IF EOF
           000
1368 E1EDA 112         A=R2
1369 E1EDD 132         AD0EX                   GET DATA STK PTR INTO D0
1370 E1EE0 184         D0=D0-  5               POINT TO LINE#
1371 E1EE3 146         C=DAT0  A               C(A)=DESIRED LINE#
1372 E1EE6 1BD9        D0=(5)  =oLINE#
           BF2
1373 E1EED 144         DAT0=C  A               SET LINE# TO DESIRED LINE
1374 E1EF0 D0          A=0     A
1375 E1EF2 1593        DAT1=A  4               WRITE 2 BYTES OF 0 FOR BYTE COUNT
1376 E1EF6 173         D1=D1+  4               PT TO DATA AREA IN BUFFER
1377
1378           *
1379           * END:  CLEAN UP & WRITE 2 NULLS TO END OF LINE
1380           *
1381
1382 E1EF9 D0          A=0     A
1383 E1EFB 1593        DAT1=A  4               WRITE 2 NULLS TO BUFFER
1384 E1EFF 03          RTNCC                   RTN W/ CARRY CLEAR, FAILED
1385
1386 E1F01 A1C  RREC5  A=A-1   WP              RESTORE BYTE COUNT
1387 E1F04 D3          D=0     A               ODD BYTE FLAG: SET FOR EVEN
1388 E1F06 D1          B=0     A
1389 E1F08 A98         B=A     WP              B(A)=REC LENGTH
1390 E1F0B 20          P=      0
1391 E1F0D 301         LC(1)   1               MASK FOR EVEN REC LEN TEST
1392 E1F10 0E05        C=B&C   P
1393 E1F14 90A         ?C=0    P               EVEN LENGTH?
1394 E1F17 40          GOYES   RREC10          YES
1395 E1F19 E7          D=D+1   A               INCREMENT FOR ODD TRAILING BYTE
1396
1397           *
1398           * NOTE: WE WRITE THE ACTUAL RECORD BYTE COUNT TO THE BUFFER
1399           * INSTEAD OF THE ACTUAL BYTE COUNT READ.
1400           *
1401
1402 E1F1B D9   RREC10 C=B     A               COPY BYTE COUNT TO C(A)
1403 E1F1D 15D3        DAT1=C  4               WRITE BYTE COUNT TO BUFFER
1404 E1F21 173         D1=D1+  4               PT TO DATA AREA IN BUFFER
1405 E1F24 02          RTNSC                   return w/ carry set for success
1406
```

```
1407                    EJECT
1408                ******************************************************
1409                *
1410                * EOF  ( --- f)  This routine examines the record length
1411                *   of the next record in the file accessed by the fib#
1412                *   in SCRFIB.  It assumes that the current pointer into
1413                *   the file is pointing at the next record length and
1414                *   that the file is a LIF1 type.
1415                *
1416                *   It returns a true if it is positioned at the end
1417                *   of file, else a false flag.
1418                *
1419                ************************************************************
1420 E1F26 F3D1             CON(5)  =!MAX
       E
1421 E1F2B 38     =!EOF     CON(2) #83            EOF
1422 E1F2D 54F4             NIBASC \EO\
1423 E1F31 6C               CON(2) \F\+#80
1424 E1F33 83F1 =EOF        CON(5)  =$EOF
       E
1425
1426 E1F38 8F00 =$EOF      GOSBVL  =SAVEFP         SAVE FORTH POINTERS
       000
1427 E1F3F 1B16             D0=(5)  =oSCFIB         GET fib#
       BF2
1428 E1F46 142              A=DAT0  A               ACTIVE FIB#
1429 E1F49 8F00             GOSBVL  =FIBAD-         GET START ADDR FOR fib
       000
1430 E1F50 8E00             GOSUBL  =GETPTR         set up cpu regs with fib pointers
       00
1431 E1F56 D2               C=0     A               D(A)=# bytes to EOF, if there are
1432 E1F58 E6               C=C+1   A               less than 2 we'll call it EOF
1433 E1F5A E6               C=C+1   A               otherwise we'll check for FFFF
1434 E1F5C 8B7              ?C>D    A               EOF marker
1435 E1F5F 02               GOYES   TRU             we are at EOF
1436
1437 E1F61 110              A=R0
1438 E1F64 130              D0=A                    D0--->INTO FILE
1439 E1F67 8F00             GOSBVL  =RDLNAS         read record length
       000
1440 E1F6E 23               P=      3
1441 E1F70 B14              A=A+1   WP              ADD 1 TEST FOR FFFF TO 0000 CARRY
1442 E1F73 20               P=      0
1443 E1F75 501              GONC    FALSE           NOT EOF
1444 E1F78 8F00             GOSBVL  =BACK2B         BACK UP FILE
       000
1445
1446 E1F7F D0   TRU         A=0     A
1447 E1F81 CC               A=A-1   A               EOF, RETURN TRUE FLAG
1448 E1F83 4B0              GOC     FAL1            FINISH
1449
1450 E1F86 8F00 FALSE       GOSBVL  =BACK2B         BACK UP FILE
       000
1451 E1F8D D0               A=0     A
1452
```



```
1453 E1F8F 8F00 FAL1        GOSBVL  =GETFP          RESTORE FORTH POINTERS
       000
1454 E1F96 1C4              D1=D1-  5
1455 E1F99 141              DAT1=A  A               PUSH RESULT FLAG
1456 E1F9C 03               RTNCC
```

```
1457                    EJECT
1458                *****************************************************
1459                *
1460                * POS                          ( str1 str2 -- p )
1461                * Find position of str1 in str2; zero if no match.
1462                *
1463                *****************************************************
1464
1465 E1F9E B2F1          CON(5)  =!EOF
          E
1466 E1FA3 38    =!POS   CON(2)  #83          POS
1467 E1FA5 05F4          NIBASC  \PO\
1468 E1FA9 3D            CON(2)  \S\+#80
1469 E1FAB 0BF1 =POS     CON(5)  =$POS
          E
1470
1471 E1FB0 8F00 =$POS   GOSBVL  =SAVEFP
          000
1472 E1FB7 8E00          GOSUBL  =SETSTR      put lengths & addresses in R0-R3
          00
1473 E1FBD D1            B=0     A
1474 E1FBF E5            B=B+1   A            start counter at 1
1475 E1FC1 11A           C=R2                 first get len to check for null
1476 E1FC4 8AA           ?C=0    A            reject null (which otherwise
1477 E1FC7 63            GOYES   FAIL            would get past SAME)
1478                * loop through using SAME on each substring
1479 E1FC9 11A   TRY     C=R2                    length string1
1480 E1FCC D7            D=C     A            D=     "
1481 E1FCE 118           C=R0                 length target
1482 E1FD1 8B3           ?D>C    A            target now  smaller?
1483 E1FD4 92            GOYES   FAIL
1484 E1FD6 8E00          GOSUBL  =RESET       set D0&D1 to the two strings
          00
1485 E1FDC 8E00          GOSUBL  =SAME        set carry if they match (through
          00
1486 E1FE2 4C1           GOC     MATCH          end of the shorter)
1487 E1FE5 118           C=R0
1488 E1FE8 CE            C=C-1   A            decrement target
1489 E1FEA E5            B=B+1   A            increment position
1490 E1FEC 108           R0=C
1491 E1FEF 119           C=R1
1492 E1FF2 E6            C=C+1   A            move pointer into str2
1493 E1FF4 E6            C=C+1   A            by 2 nibbles
1494 E1FF6 109           R1=C
1495 E1FF9 6FCF          GOTO    TRY
1496
1497 E1FFD       =fFAIL
1498 E1FFD D1    FAIL    B=0     A            false
1499
1500 E1FFF       =fMATCH
1501 E1FFF D4    MATCH   A=B     A            save from GETFP
1502 E2001 8F00          GOSBVL  =GETFP
          000
1503 E2008 17E           D1=D1+  15           4 in, 1 out
1504 E200B 141           DAT1=A  A
```



```
1505 E200E 03            RTNCC
```

```
1506                    EJECT
1507                ****************************************
1508                *
1509                * VAL                       ( str -- d )
1510                * Return number equivalent of string.
1511                *
1512                ****************************************
1513
1514 E2010 3AF1            CON(5)  =!POS
           E
1515 E2015 38      =!VAL   CON(2)  #83          VAL
1516 E2017 6514            NIBASC  \VA\
1517 E201B CC              CON(2)  \L\+#80
1518
1519 E201D 0000 =VAL       CON(5)  =DOCOL       :
           0
1520 E2022 0C42            CON(5)  =DROP        DROP
           E
1521 E2027 0000            CON(5)  =TWO-        2-
           0
1522
1523                * drop count and point to count behind str
1524
1525 E202C 0000            CON(5)  =NUMB        let NUMBER translate string
           0
1526 E2031 0000            CON(5)  =SEMI        ;
           0
1527
1528                ****************************************
1529                *
1530                * ASC                       ( str -- ascii )
1531                * Give ascii value of first char of string.
1532                *
1533                *
1534                ****************************************
1535
1536 E2036 5102            CON(5)  =!VAL
           E
1537 E203B 38      =!ASC   CON(2)  #83
1538 E203D 1435            NIBASC  \AS\
1539 E2041 3C              CON(2)  \C\+#80
1540
1541 E2043 0000 =ASC       CON(5)  =DOCOL       :
           0
1542 E2048 0C42            CON(5)  =DROP        drop len
           E
1543 E204D 0000            CON(5)  =CAT         fetch 1st char
           0
1544 E2052 0000            CON(5)  =SEMI        ;
           0
```



```
1545                    $TITLE  4 CHAR WORDS
1546                ****************************************
1547                *                                      *
1548                *       START OF 4 CHAR WORDS          *
1549                *   ADD ADDITIONAL WORDS AT THIS END    *
1550                *                                      *
1551                ****************************************
1552
1553                ****************************************
1554                *                                      *
1555                * FABS                                 *
1556                *                                      *
1557                * REPLACE X REGISTER BY ABSOLUTE VALUE OF X *
1558                *                                      *
1559                ****************************************
1560 E2057 0000              CON(5)  0
           0
1561 E205C 48      =!FABS    CON(2)  #84
1562 E205E 6414            NIBASC  \FAB\
           24
1563 E2064 3D              CON(2)  \S\+#80
1564 E2066 B602 =FABS      CON(5)  =$FABS
           E
1565 E206B 340D =$FABS     LC(5)   =oX
           BF2
1566
1567                * PUT X IN LASTX (CAN'T CALL GETX+L BECAUSE OF
1568                * SIGNAL NAN'S WHICH MUST NOT BE CHECKED)
1569
1570 E2072 136               CD0EX
1571 E2075 1527            A=DAT0  W           GET X
1572 E2079 18F             D0=D0-  16          LASTX
1573 E207C 1507            DAT0=A  W           REPLACE WITH X
1574 E2080 16F             D0=D0+  16          BACK TO X
1575 E2083 16E             D0=D0+  15          POINT TO SIGN NIBBLE OF X
1576 E2086 AC0             A=0     S
1577 E2089 1504            DAT0=A  S           PUT IN 0 FOR POSITIVE
1578 E208D 134             D0=C                RESTORE D0
1579 E2090 03              RTNCC
1580                ****************************************
1581                *
1582                * CRLF   ( --- str)  This word returns
1583                *       a pointer to a string constant
1584                *       (constant because its in ROM) of
1585                *       a carriage return and linefeed.
1586                *       Handy for use with OUTPUT.
1587                *
1588                ****************************************
1589 E2092 C502            CON(5)  =!FABS
           E
1590 E2097 48      =!CRLF  CON(2)  #84
1591 E2099 3425            NIBASC  \CRL\
           C4
1592 E209F 6C              CON(2)  \F\+#80
1593 E20A1 0000 =CRLF      CON(5)  =DOCOL       :
```

```
          0
1594 E20A6 0000        CON(5)  =QUOTC      string runtime code
          0
1595 E20AB 20          CON(2)  2           max len
1596 E20AD 20          CON(2)  2           current len
1597 E20AF D0          CON(2)  13          carriage return
1598 E20B1 A0          CON(2)  10          line feed
1599 E20B3 0000        CON(5)  =SEMI       ;
          0
```



```
1600                    EJECT
1601              ********************************************
1602              *
1603              *  X=Y?  X>Y?  X=0?  X<Y?
1604              *
1605              *  Stack:  --- t/f
1606              *
1607              *  Perform the indicated test on X and Y of the
1608              *  floating point stack, if true push -1 to integer
1609              *  stack, else push 0 to integer stack.
1610              *
1611              ************************************************
1612 E20B8 7902         CON(5)  !CRLF
          E
1613 E20BD 48    =!X<Y? CON(2)  #84
1614 E20BF 85C3         NIBASC  \X<Y\
          95
1615 E20C5 FB           CON(2)  \?\+#80
1616 E20C7 CC02 =X<Y?   CON(5)  $X<Y?
          E
1617 E20CC 7990 $X<Y?   GOSUB   CMPSI-
1618 E20D0 21           P=      1           INDICATE OPERATION
1619 E20D2 6A50         GOTO    XXYY        FINISH THERE
1620
1621 E20D6 DB02         CON(5)  !X<Y?
          E
1622 E20DB 48    =!XE0? CON(2)  #84
1623 E20DD 85D3         NIBASC  \X=0\
          03
1624 E20E3 FB           CON(2)  \?\+#80
1625 E20E5 AE02 =XE0?   CON(5)  $XE0?
          E
1626 E20EA 850   $XE0?  ST=1    0
1627 E20ED 7B70         GOSUB   CMPST
1628              *     C=0     W           SET C TO 0
1629 E20F1 6930         GOTO    X1YY        FINISH THERE
1630
1631 E20F5 BD02         CON(5)  !XE0?
          E
1632 E20FA 48    =!X>Y? CON(2)  #84
1633 E20FC 85E3         NIBASC  \X>Y\
          95
1634 E2102 FB           CON(2)  \?\+#80
1635 E2104 9012 =X>Y?   CON(5)  $X>Y?
          E
1636 E2109 7C50 $X>Y?   GOSUB   CMPSI-
1637 E210D 24           P=      4           P INDICATES TYPE OF TEST
1638 E210F 6D10         GOTO    XXYY        FINISH THERE
1639
1640 E2113 AF02         CON(5)  =!X>Y?
          E
1641 E2118 48    =!XEY? CON(2)  #84
1642 E211A 85D3         NIBASC  \X=Y\
          95
1643 E2120 FB           CON(2)  \?\+#80
```

```
1644 E2122 7212 =XEY?   CON(5)  $XEY?
           E
1645 E2127 7E30 =$XEY?  GOSUB   CMPST-
1646 E212B 22   X1YY    P=      2              P INDICATES TYPE OF TEST
1647 E212D 8F00 =XXYY   GOSBVL  =uTEST
           000
1648 E2134 04           SETHEX
1649 E2136 D0           A=0     A              RETURN FALSE FLAG
1650 E2138 540          GONC    TR0
1651 E213B CC           A=A-1   A              TRUE FLAG
1652 E213D 8E00 TR0     GOSUBL  =GETFP         RESTORE FORTH ENVIRONMENT
           00
1653 E2143 1C4          D1=D1-  5
1654 E2146 141          DAT1=A  A              PUSH FLAG TO STACK
1655 E2149 03           RTNCC
1656
1657 E214B 8112         CON(5)  =!XEY?
           E
1658 E2150 48   =!X#Y?  CON(2)  #84
1659 E2152 8532         NIBASC  \X#Y\
           95
1660 E2158 FB           CON(2)  \?\+#80
1661 E215A F512 =X#Y?   CON(5)  =$X#Y?
           E
1662 E215F 7600 =$X#Y?  GOSUB   CMPST-
1663 E2163 2D           P=      13
1664 E2165 67CF         GOTO    XXYY
1665
1666            ************************************************
1667            *
1668            * CMPST:        Startup routine for comparison
1669            *               operators
1670            *               x=y? x=0? x<y? x#y? x<=y? x>=y? x>y?
1671            *
1672            * Entry: ST[0]=0 iff want both X and Y (i.e.not
1673            *        doing x=0?)
1674            *
1675            * Exit: If ST[0]=0 then A=X (packed) and C=Y (packed)
1676            *       If ST[0]=1 then A=X (packed) and C=0
1677            *       Signaling NaN's reported and replaced
1678            *       FORTH pointers saved
1679            *       D1=[MTHSTK]
1680            *
1681            * Stk levels: 4 max
1682            * Uses: A,B,C,D,P,R0,R3,D0,D1,ST[7..11]
1683            ************************************************
1684
1685 E2169 840  =CMPST- ST=0    0              Not doing X=0?
1686 E216C 8E00 =CMPST  GOSUBL  =SAVEFP
           00
1687 E2172 05           SETDEC
1688 E2174 1F00         D1=(5)  =MTHSTK
           000
1689 E217B 147          C=DAT1  A              C=[MTHSTK]
1690 E217E 135          D1=C                   SET D1 TO MTHSTK FOR ERRORS
```



```
1691 E2181 1B0D         D0=(5)  (=oX)
           BF2
1692 E2188 7C10         GOSUB   CMPSUB         A=X
1693 E218C 860          ?ST=0   0              DOING X=0?
1694 E218F 70           GOYES   CMPST+         JUMP IF NOT
1695 E2191 AF2          C=0     W              SET UP FOR X=0?
1696 E2194 01           RTN
1697 E2196 100  CMPST+  R0=A                   R0=X
1698 E2199 16F          D0=D0+  16             D0-->Y
1699 E219C 7800         GOSUB   CMPSUB         A=Y
1700 E21A0 AF6          C=A     W              C=Y
1701 E21A3 110          A=R0                   A=X
1702 E21A6 01           RTN
1703
1704 E21A8 1527 CMPSUB  A=DAT0  W              A=X OR Y
1705 E21AC 8F00         GOSBVL  =SIGTST        CHECK SIGNALING NAN'S
           000
1706 E21B3 500          RTNNC                  NO S.N.
1707 E21B6 8F00         GOSBVL  =uRES12        PACK UP S.N.
           000
1708 E21BD AFA          A=C     W
1709 E21C0 01           RTN
```

```
1710                    EJECT
1711                *****************************************
1712                *
1713                * SQRT          SQUARE ROOT OF X
1714                *
1715                *****************************************
1716 E21C2 0512            CON(5)  !X*Y?
          E
1717 E21C7 48      *!SQRTX CON(2)  #84
1718 E21C9 3515            NIBASC  \SQR\
          25
1719 E21CF 4D              CON(2)  \T\+#80
1720 E21D1 6D12    *SQRTX  CON(5)  *$SQRTX
          E
1721 E21D6 8EC3    *$SQRTX GOSUBL  *NUMST      SAVEFP;GETX+L
          5F
1722 E21DC 8F00            GOSBVL  *SQR15
          000
1723 E21E3 8C00            GOLONG  *FEND
          00
```



```
1724                    EJECT
1725                *****************************************
1726                *
1727                * ITOF    ( n --- )  This routine pops the
1728                *         top value on the FORTH data stack
1729                *         and pushes it onto the floating point
1730                *         stack.
1731                *
1732                *****************************************
1733 E21E9 7C12              CON(5) !SQRTX         NO MORE 4 CHAR WORDS
          E
1734 E21EE 48      =!ITOF   CON(2) #84
1735 E21F0 9445             NIBASC \ITO\
          F4
1736 E21F6 6C               CON(2) \F\+#80
1737 E21F8 DF12    =ITOF    CON(5) $ITOF
          E
1738
1739 E21FD 8ED5    $ITOF    GOSUBL =$DUP          DUP VALUE
          7F
1740 E2203 8EA1             GOSUBL =$ABS          MAKE COPY |n|
          8F
1741 E2209 143              A=DAT1 A              READ VALUE
1742 E220C 174              D1=D1+ 5
1743 E220F 8E00             GOSUBL =SAVEFP        SAVE FORTH ENVIRONMENT
          00
1744 E2215 DC               ABEX   A              SAVE IN B(A)
1745 E2217 8E00             GOSUBL =STKLFT        PREPARE FOR F.P. PUSH
          00
1746 E221D DC               ABEX   A              RESTORE TO A(A)
1747 E221F 8F00             GOSBVL =HDFLT         CHANGE TO FLOATING POINT
          000
1748 E2226 04               SETHEX                RESTORE HEX MODE
1749 E2228 8E00             GOSUBL =GETFP         RESTORE FORTH POINTERS
          00
1750 E222E 147              C=DAT1 A              GET ORIGINAL #
1751 E2231 1B0D             D0=(5) =oX            SET UP FOR WRITE TO X
          BF2
1752 E2238 C6               C=C+C  A
1753 E223A 590              GONC   ITOF0          # WAS POSITIVE, AOKAY
1754 E223D 05               SETDEC
1755 E223F A4C              A=A-1  S              MAKE IT NEGATIVE
1756 E2242 04               SETHEX
1757 E2244 1507 ITOF0       DAT0=A W              WRITE TO F.P. STACK
1758 E2248 8E00             GOSUBL =GETFP         RESTORE FORTH & RTN
          00
1759 E224E 174              D1=D1+ 5              OFFICIALLY DROP COPY
1760 E2251 03               RTNCC
```

```
1761                    EJECT
1762                ********************************************
1763                *
1764                * FTOI  ( --- n)  This routine copies the
1765                *       value in the floating point X register
1766                *       and pushes it to the FORTH data stack.
1767                *
1768                ********************************************
1769 E2253 EE12E           CON(5) !ITOF
1770 E2258 48    =!FTOI    CON(2) #84
1771 E225A 6445F4          NIBASC \FTO\
1772 E2260 9C              CON(2) \I\+#80
1773 E2262 7622E  =FTOI    CON(5) $FTOI
1774
1775 E2267 8E0000 $FTOI    GOSUBL =SAVEFP          SAVE FORTH ENVIRONMENT
1776 E226D 1B0DBF2         D0=(5) =oX
1777 E2274 1527            A=DAT0 W                GET VALUE OF X REG
1778 E2278 8E0000          GOSUBL =FLTOH           CHANGE TO INTEGER
1779                *                               ERROR OUT IF OVERFLOW
1780 E227E 8E0000          GOSUBL =GETFP           RESTORE FORTH POINTERS
1781 E2284 1C4             D1=D1- 5
1782 E2287 141             DAT1=A A                PUSH NEW INTEGER
1783 E228A 03              RTNCC
```



```
1784                    EJECT
1785                ********************************************
1786                *
1787                * #TIB  ( --- addr)  This rtne returns the
1788                *       addr of a variable which contains
1789                *       the number of characters in the
1790                *       TIB.  It is set by Query.
1791                *
1792                ********************************************
1793 E228C 8522E           CON(5) !FTOI
1794 E2291 48    =!#TIB    CON(2) #84
1795 E2293 324594          NIBASC \#TI\
1796 E2299 2C              CON(2) \B\+#80
1797 E229B 0A22E  =#TIB    CON(5) $#TIB
1798
1799 E22A0 8E0000 =$#TIB   GOSUBL =DOUSE
1800 E22A6 E3BF2           CON(5) (=o#TIB)
```

```
1801                    EJECT
1802              ***************************************
1803              *
1804              * SPAN  ( --- addr )  This rtne returns the
1805              *       addr of a variable which contains
1806              *       the number of characters read during
1807              *       the last execution of EXPECT96.
1808              *
1809              ***************************************
1810 E22AB 1922           CON(5)  !#TIB
          E
1811 E22B0 48     =!SPAN  CON(2)  #84
1812 E22B2 3505           NIBASC  \SPA\
          14
1813 E22B6 EC             CON(2)  \N\+#80
1814 E22BA FB22   =SPAN   CON(5)  $SPAN
          E
1815
1816 E22BF 8E00 $SPAN     GOSUBL  =DOUSE
          00
1817 E22C5 C5BF           CON(5)  (=aSPAN)
          2
```



```
1818                    EJECT
1819              ***************************************
1820              *
1821              * EXIT  ( --- )  Terminates execution of
1822              *       a colon-definition.  May not be
1823              *       used within a DO--LOOP
1824              *
1825              ***************************************
1826 E22CA 0B22           CON(5) !SPAN
          E
1827 E22CF 48     =!EXIT  CON(2) #84
1828 E22D1 5485           NIBASC \EXI\
          94
1829 E22D7 4D             CON(2) \T\+#80
1830 E22D9 0000 =EXXIT   CON(5) =DOCOL          :
          0
1831 E22DE 0000           CON(5) =R>
          0
1832 E22E3 0C42           CON(5) =DROP
          E
1833 E22E8 0000           CON(5) =SEMI          ;
          0
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER

```
1834                    EJECT
1835                ******************************************
1836                *
1837                * FLIT    ( --- )  Pushes the next 16 nibs
1838                *         from the dictionary onto the
1839                *         floating point stack.
1840                *
1841                *         This word is a headerless
1842                *         primitive.
1843                *
1844                ******************************************
1845
1846 E22ED 2F22 =FLIT    CON(5) $FLIT
           E
1847
1848 E22F2 1527 $FLIT    A=DAT0 W              READ F.P. VALUE
1849 E22F6 100           R0=A                  SAVE IT
1850 E22F9 16F           D0=D0+ 16             UPDATE I
1851 E22FC 8E00          GOSUBL =SAVEFP        SAVE FORTH POINTERS
           00
1852 E2302 8E00          GOSUBL =STKLFT        READY FOR STORE
           00
1853 E2308 1B0D          D0=(5) =oX
           BF2
1854 E230F 110           A=R0
1855 E2312 1507          DAT0=A W              STORE VALUE IN X
1856 E2316 8C00          GOLONG =GETFP         RECOVER FORTH POINTERS
           00
1857
1858                ******************************************
1859                *
1860                * ASIN          ARC-SINE
1861                *
1862                ******************************************
1863 E231C FC22          CON(5)  !EXIT         NO MORE 4 CHAR WORDS
           E
1864 E2321 48    =!ASIN  CON(2)  #84
1865 E2323 1435          NIBASC  \ASI\
           94
1866 E2329 EC            CON(2)  \N\+#80
1867 E232B 0332  =ASIN   CON(5)  =$ASIN
           E
1868 E2330 8E2E  =$ASIN  GOSUBL  =NUMST     SAVEFP;GETX+L;UMODES
           3F
1869 E2336 8F00          GOSBVL  =ASIN15
           000
1870 E233D 8C00          GOLONG  =FEND       PUTABX;GETFP;RTNCC
           00
1871
1872                ******************************************
1873                *
1874                * ACOS          ARC-COSINE
1875                *
1876                ******************************************
1877 E2343 1232          CON(5)  =!ASIN
```



```
           E
1878 E2348 48    =!ACOS  CON(2)  #84
1879 E234A 1434          NIBASC  \ACO\
           F4
1880 E2350 3D            CON(2)  \S\+#80
1881 E2352 7532  =ACOS   CON(5)  =$ACOS
           E
1882 E2357 8EBB  =$ACOS  GOSUBL  =NUMST
           3F
1883 E235D 8F00          GOSBVL  =ACOS15
           000
1884 E2364 8C00          GOLONG  =FEND
           00
1885
1886                ******************************************
1887                *
1888                * ATAN          ARC-TANGENT
1889                *
1890                ******************************************
1891 E236A 8432          CON(5)  =!ACOS
           E
1892 E236F 48    =!ATAN  CON(2)  #84
1893 E2371 1445          NIBASC  \ATA\
           14
1894 E2377 EC            CON(2)  \N\+#80
1895 E2379 E732  =ATAN   CON(5)  =$ATAN
           E
1896 E237E 8E49  =$ATAN  GOSUBL  =NUMST
           3F
1897 E2384 8F00          GOSBVL  =ATAN15
           000
1898 E238B 8C00          GOLONG  =FEND
           00
1899
1900                ******************************************
1901                *
1902                * X<>Y          EXCHANGE X & Y  41C REGISTERS
1903                *
1904                ******************************************
1905 E2391 F632          CON(5)  =!ATAN
           E
1906 E2396 48    =!X<>Y  CON(2)  #84
1907 E2398 85C3          NIBASC  \X<>\
           E3
1908 E239E 9D            CON(2)  \Y\+#80
1909 E23A0 5A32  =X<>Y   CON(5)  =$X<>Y
           E
1910 E23A5 20    =$X<>Y  P=      0
1911 E23A7 340D          LC(5)   =oX
           BF2
1912 E23AE 136           CD0EX
1913 E23B1 109           R1=C                  SAVE D0
1914 E23B4 1567          C=DAT0  W             C=X
1915 E23B8 16F           D0=D0+  16            D0-->Y
1916 E23BB 1527          A=DAT0  W             A=Y
```

```
1917 E23BF 1547          DAT0=C  W               Y=X
1918 E23C3 18F           D0=D0-  16              D0-->X
1919 E23C6 1507          DAT0=A  W               X=Y
1920 E23CA 119           C=R1
1921 E23CD 134           D0=C                    RESTORE D0
1922 E23D0 03            RTNCC
1923                     ****************************************
1924                     *
1925                     * 10^X          10 TO THE XTH POWER
1926                     *
1927                     ****************************************
1928 E23D2 6932          CON(5)  =!X<>Y
           E
1929 E23D7 48    =!10^X  CON(2)  #84
1930 E23D9 1303          NIBASC  \10^\
           E5
1931 E23DF 8D            CON(2)  \X\+#80
1932 E23E1 6E32  =10^X   CON(5)  =$10^X
           E
1933 E23E6 8EC2  =$10^X  GOSUBL  =NUMST       GET X INTO A(A,B), STORE IN LASTX
           3F
1934 E23EC AF9           C=B     W
1935 E23EF AF7           D=C     W
1936 E23F2 AF6           C=A     W              COPY X INTO (C,D)
1937 E23F5 AF0           A=0     W
1938 E23F8 E4            A=A+1   A
1939 E23FA AF1           B=0     W
1940 E23FD 2E            P=      14
1941 E23FF B05           B=B+1   P              (A,B)=10
1942 E2402 8F00          GOSBVL  =YX2-15        DO FUNCTION
           000
1943 E2409 8C00          GOLONG  =FEND
           00
```



```
1944                     EJECT
1945                     ****************************************************
1946                     *
1947                     * GROW  ( n --- t/f)  Enlarge the user dictionary
1948                     *        by n nibs.  If there is enough room, return
1949                     *        a true flag (-1) else return a false flag (0).
1950                     *
1951                     * changed n to mean nibs instead of bytes
1952                     * 12/12/83 B.M.
1953                     ****************************************************
1954
1955
1956 E240F 7D32              CON(5) =!10^X
           E
1957 E2414 48    =!GROW      CON(2) #84
1958 E2416 7425              NIBASC \GRO\
           F4
1959 E241C 7D                CON(2) \W\+#80
1960 E241E 3242  =GROW       CON(5) $GROW
           E
1961
1962 E2423 7360  =$GROW      GOSUB  CHK          make sure n is legal
1963 E2427 06                RSTK=C              save #NIBS on CPU stk
1964 E2429 8F00              GOSBVL =MEMCKL      check free mem, leaves
           000
1965                         *                   amount to check in B(A)
1966 E2430 417               GOC    WRO          not enough, rtn fail flag
1967 E2433 1B00              D0=(5) =MAINST
           000
1968 E243A 146               C=DAT0 A            C(A)=addr of file
1969 E243D 1B39              D0=(5) =oDP
           BF2
1970 E2444 142               A=DAT0 A            A(A)=addr of 1st free nib
1971
1972 E2447 8F00              GOSBVL =MVMEM+      move memory, adjust file chain
           000
1973
1974 E244E 07    =adj        C=RSTK              fetch #nibs from cpu rtn stack
1975 E2450 1B11              D0=(5) =oS0
           BF2
1976 E2457 DA                A=C    A            A(A)=# NIBS change in dictionary
1977 E2459 3160              LC(2)  6            7 entries (7-1) to adjust
1978 E245D AE5               B=C    B            store counter in B(B)
1979
1980 E2460 146   AD1         C=DAT0 A            read value
1981 E2463 C2                C=C+A  A            adjust value
1982 E2465 144               DAT0=C A            write new value
1983 E2468 164               D0=D0+ 5            adjust ptr to next value
1984 E246B A6D               B=B-1  B            decrement counter
1985 E246E 51F               GONC   AD1          we're done on carry
1986
1987 E2471 8E00              GOSUBL =GETFP       restore FORTH pointers
           00
1988
1989                     * must also adjust data stk and rtn stk, A(A) still
```

```
1990            * has #NIBS adjusted after restoring FORTH pointers
1991
1992 E2477 C8              B=B+A  A            add amount to RTN STK
1993 E2479 137             CD1EX
1994 E247C C2              C=C+A  A            add amount to DATA STK
1995 E247E 135             D1=C
1996 E2481 D2              C=0    A
1997 E2483 CE              C=C-1  A            true flag
1998 E2485 145             DAT1=C A            push flag
1999 E2488 03              RTNCC
2000
2001 E248A 8E00 =CHK       GOSUBL =SAVEFP      save FORTH pointers
          00
2002 E2490 20              P=     0
2003 E2492 AF2             C=0    W
2004 E2495 147             C=DAT1 A            read # nibs
2005 E2498 C6              C=C+C  A            check for negative #s
2006 E249A 470             GOC    WR0          error if negative
2007 E249D 81E             CSRB                restore to nib count
2008 E24A0 01              RTN
2009
2010 E24A2 07   WR0        C=RSTK              throw away CHK rtn stk
2011
2012 E24A4 8E00 =WR1       GOSUBL =GETFP       restore FORTH pointers
          00
2013 E24AA D2              C=0    A
2014 E24AC 145             DAT1=C A            write false flag
2015 E24AF 03              RTNCC
```



```
2016                EJECT
2017            ************************************************
2018            *
2019            * DROP ( n -- )  drop the top number from the stack
2020            *
2021            ************************************************
2022
2023 E24B1 4142            CON(5)  !GROW
          E
2024 E24B6 48    =!DROP    CON(2)  #84              DROP
2025 E24B8 4425            NIBASC  \DRO\
          F4
2026 E24BE 0D              CON(2)  \P\+#80
2027 E24C0 5C42  =DROP     CON(5)  =$DROP
          E
2028
2029 E24C5 174   =$DROP    D1=D1+  5               THROW AWAY TOP ITEM
2030 E24C8 03              RTNCC
2031
2032
2033            ************************************************
2034            *
2035            * SWAP  ( n1 n2 -- n2 n1 ) exchange order of top two
2036            *                          stack items
2037            *
2038            ************************************************
2039
2040 E24CA 6B42            CON(5)  =!DROP
          E
2041 E24CF 48    =!SWAP    CON(2)  #84              SWAP
2042 E24D1 3575            NIBASC  \SWA\
          14
2043 E24D7 0D              CON(2)  \P\+#80
2044 E24D9 ED42  =SWAP     CON(5)  =$SWAP
          E
2045
2046 E24DE 147   =$SWAP    C=DAT1  A               POP N2
2047 E24E1 174             D1=D1+  5
2048 E24E4 143             A=DAT1  A               POP N1
2049 E24E7 145             DAT1=C  A               PUSH N2
2050 E24EA 1C4             D1=D1-  5
2051 E24ED 141             DAT1=A  A               PUSH N1
2052 E24F0 03              RTNCC
```

```
2053                    EJECT
2054                *************************************************************
2055                *
2056                * PICK  (n1 -- n2)  return contents of  n1-th stack value
2057                *                   not counting n1 itself. NO ERROR CHECKING!
2058                *
2059                *************************************************************
2060
2061 E24F2 FC42            CON(5)  =!SWAP
          E
2062 E24F7 48     =!PICK   CON(2)  #84             PICK
2063 E24F9 0594           NIBASC  \PIC\
          34
2064 E24FF BC             CON(2)  \K\+#80
2065 E2501 6052 =PICK     CON(5)  =$PICK
          E
2066
2067 E2506 147  =$PICK    C=DAT1  A               C=n1
2068 E2509 DA             A=C     A
2069 E250B C6             C=C+C   A               (n1+1)*2
2070 E250D C6             C=C+C   A               (n1+1)*4
2071 E250F C2             C=C+A   A               (n1+1)*5=OFFSET INTO STACK
2072 E2511 133            AD1EX                   SAVE DATA STK IN A
2073 E2514 C2             C=C+A   A               ADDR OF DESIRED ITEM
2074 E2516 135            D1=C
2075 E2519 147            C=DAT1  A               C=DESIRED ITEM
2076 E251C 131            D1=A                    RESTORE DATA STACK
2077 E251F 145            DAT1=C  A               STORE ITEM ON TOP OF n1
2078 E2522 03             RTNCC
```



```
2079                    EJECT
2080                *************************************************************
2081                *
2082                * OVER  (n1 n2 -- n1 n2 n1) leave a copy of 2nd number
2083                *                           on data stack
2084                *
2085                *************************************************************
2086 E2524 7F42            CON(5)  =!PICK
          E
2087 E2529 48     =!OVER   CON(2)  #84             OVER
2088 E252B F465           NIBASC  \OVE\
          54
2089 E2531 2D             CON(2)  \R\+#80
2090 E2533 8352 =OVER     CON(5)  =$OVER
          E
2091
2092 E2538 174  =$OVER    D1=D1+  5
2093 E253B 147            C=DAT1  A               POP n1
2094 E253E 1C9            D1=D1-  10
2095 E2541 145            DAT1=C  A               PUSH n1
2096 E2544 03             RTNCC
2097
2098                *************************************************************
2099                *
2100                * 2DUP  (d --- d d) duplicate top double # on data stack
2101                *
2102                *************************************************************
2103
2104 E2546 9252            CON(5)  =!OVER
          E
2105 E254B 48     =!2DUP   CON(2)  #84             2DUP
2106 E254D 2344           NIBASC  \2DU\
          55
2107 E2553 0D             CON(2)  \P\+#80
2108 E2555 A552 =2DUP     CON(5)  =$2DUP
          E
2109
2110 E255A 15F9 =$2DUP    C=DAT1  10              POP D
2111 E255E 1C9            D1=D1-  10
2112 E2561 15D9           DAT1=C  10              STORE D AGAIN
2113 E2565 03             RTNCC
```

```
2114                    EJECT
2115                ************************************************************
2116                *
2117                * FILL (addr u byte --- ) fill memory beginning at addr with
2118                *       n copies of byte, if quantity is  = 0 then
2119                *       take no action
2120                *
2121                ************************************************************
2122
2123 E2567 B452            CON(5)  =!2DUP
             E
2124 E256C 48     =!FILL    CON(2)  #84             FILL
2125 E256E 6494            NIBASC  \FIL\
             C4
2126 E2574 CC              CON(2)  \L\+#80
2127 E2576 B752 =FILL      CON(5)  =$FILL
             E
2128
2129 E257B 143  =$FILL     A=DAT1  A               READ BYTE TO A
2130 E257E 174             D1=D1+  5
2131 E2581 147             C=DAT1  A               READ U
2132 E2584 174             D1=D1+  5
2133 E2587 D7              D=C     A               COPY U TO D
2134 E2589 8AB             ?D=0    A               ANY TO DO?
2135 E258C 81              GOYES   FIL1            NO, LETS FINISH
2136 E258E CF              D=D-1   A               PRIME FOR CARRY TEST
2137 E2590 147             C=DAT1  A               READ ADDR
2138 E2593 136             CD0EX                   D0=ADDR,C=I
2139 E2596 148  FIL0       DAT0=A  B               STORE BYTE AT ADDR
2140 E2599 161             D0=D0+  2               INCREMENT ADDR
2141 E259C CF              D=D-1   A               DECREMENT COUNT
2142 E259E 57F             GONC    FIL0            DO ANOTHER IF NO CARRY
2143 E25A1 134             D0=C                    RESTORE I TO D0
2144 E25A4 174  FIL1       D1=D1+  5               DROP ADDR FROM DATA STACK
2145 E25A7 03              RTNCC
```



```
2146                    EJECT
2147                ************************************************************
2148                *
2149                * DABS  (d --- |d| ) absolute value of dbl # on data stack
2150                *
2151                ************************************************************
2152
2153 E25A9 C652              CON(5)  =!FILL
             E
2154 E25AE 48     =!DABS    CON(2)  #84             DABS
2155 E25B0 4414            NIBASC  \DAB\
             24
2156 E25B6 3D              CON(2)  \S\+#80
2157 E25B8 DB52 =DABS      CON(5)  =$DABS
             E
2158
2159 E25BD 143  =$DABS     A=DAT1  A               get MSDs for quick test
2160 E25C0 C4              A=A+A   A
2161 E25C2 500             RTNNC
2162
2163                * now we know it's negative
2164
2165 E25C5 1C4             D1=D1-  5
2166 E25C8 15B9            A=DAT1  10              get MSDs
2167 E25CC 179             D1=D1+  10
2168 E25CF 143             A=DAT1  A               get LSDs
2169 E25D2 29              P=      9
2170 E25D4 B98             A=-A    WP
2171 E25D7 141             DAT1=A  A               put LSDs
2172 E25DA 1C9             D1=D1-  10
2173 E25DD 1599            DAT1=A  10              put MSDs
2174 E25E1 174             D1=D1+  5
2175 E25E4 03              RTNCC
```

```
2176                    EJECT
2177              ******************************************************
2178              *
2179              * S->D  (n --- d)  sign extend a single # to form a dbl #
2180              *
2181              ******************************************************
2182
2183 E25E6 EA52           CON(5)  =!DABS
         E
2184 E25EB 48     =!S->D  CON(2)  #84             S->D
2185 E25ED 35D2           NIBASC  \S->\
         E3
2186 E25F3 4C             CON(2)  \D\+#80
2187 E25F5 AF52 =S->D     CON(5)  =$S->D
         E
2188
2189 E25FA 147  =$S->D    C=DAT1  A               POP N
2190 E25FD D0             A=0     A               CLEAR FOR POS #
2191 E25FF C6             C=C+C   A               CHECK SIGN BIT
2192 E2601 540            GONC    SD0             GO IF POSITIVE
2193 E2604 CC             A=A-1   A               ITS NEG MAKE ALL 1'S
2194 E2606 1C4  SD0       D1=D1-  5
2195 E2609 141            DAT1=A  A               PUSH SIGN EXTENSION
2196 E260C 03             RTNCC
2197
2198              ******************************************************
2199              *
2200              * WARN  ( --- addr)  addr of variable with warning
2201              *       flag
2202              *
2203              ******************************************************
2204
2205 E260E BE52           CON(5)  =!S->D
         E
2206 E2613 48     =!WARN  CON(2)  #84             WARN
2207 E2615 7514           NIBASC  \WAR\
         25
2208 E261B EC             CON(2)  \N\+#80
2209 E261D 2262 =WARN     CON(5)  =$WARN
         E
2210
2211 E2622 8E00 =$WARN    GOSUBL  =DOUSE
         00
2212 E2628 84BF           CON(5)   (=oWARN)
         2
2213
2214              ******************************************
2215              *
2216              * PREV  ( --- addr)  addr of variable having
2217              *       addr of last used disc buffer.
2218              *
2219              ******************************************
2220
2221 E262D 3162           CON(5)  =!WARN
         E
```



```
2222 E2632 48     =!PREV  CON(2)  #84             PREV
2223 E2634 0525           NIBASC  \PRE\
         54
2224 E263A 6D             CON(2)  \V\+#80
2225 E263C 1462 =PREV     CON(5)  =$PREV
         E
2226
2227 E2641 8E00 =$PREV    GOSUBL   =DOUSE
         00
2228 E2647 52BF           CON(5)   (=oPREV)
         2
```

```
2229                    EJECT
2230              ****************************************
2231              *
2232              * BASE  ( --- addr)  addr of variable
2233              *       containing the currently active
2234              *       base for interpreting and displaying
2235              *       numbers
2236              *
2237              ********************************************
2238
2239 E264C 2362          CON(5)  =!PREV
          E
2240 E2651 48    =!BASE   CON(2)  #84             BASE
2241 E2653 2414          NIBASC  \BAS\
          35
2242 E2659 5C            CON(2)  \E\+#80
2243 E265B 0662 =BASE    CON(5)  =$BASE
          E
2244
2245 E2660 8E00 =$BASE   GOSUBL   =DOUSE
          00
2246 E2666 57BF          CON(5)   (=oBASE)
          2
```



```
2247                    EJECT
2248              ************************************
2249              *
2250              * THEN   compile time: ( addr 2 - )
2251              * no run-time action
2252              *
2253              * Check for syntax (i.e. that there was an IF as preceding
2254              * control=structure word; then fill in the offset from
2255              * here back to the IF after the ZBRNH (conditional jump)
2256              * which IF compiled.
2257              *
2258              ************************************
2259
2260 E266B 1562          CON(5)  =!BASE
          E
2261 E2670 4C    =!THEN   CON(2) #C4              THEN
2262 E2672 4584          NIBASC \THE\
          54
2263 E2678 EC            CON(2) \N\+#80
2264
2265 E267A 0000 =THEN    CON(5) =DOCOL
          0
2266 E267F 0000          CON(5)  =?COMP          ?COMP
          0
2267 E2684 0000          CON(5)  =TWO            2
          0
2268 E2689 0000          CON(5)  =?PAIR          ?PAIRS
          0
2269 E268E 95B2          CON(5)  =HERE           HERE
          E
2270 E2693 3352          CON(5)  =OVER          get addr of the IF which it left und
          E
2271 E2698 0000          CON(5)  =MINUS         get offset back from HERE to there
          0
2272 E269D 9D42          CON(5)  =SWAP          stack: offset addr
          E
2273 E26A2 0000          CON(5)  =STORE        store offset back at IF, i.e. after
          0
2274              *                             the ZBRNH which it compiled
2275 E26A7 0000          CON(5)  =SEMI           ;
          0
2276
2277
2278              ********************************************************
2279              *
2280              * ELSE   compile time: ( addr1 2 - addr2 2 )   immediate
2281              * compiles BRANCH ( - )
2282              *
2283              * ELSE works by compiling an unconditional branch which will
2284              * have its addr filled in by THEN (which will think that an IF
2285              * preceded it since we leave the same syntax-check code), and
2286              * executing THEN to fill in the addr for the conditional jump
2287              * compiled by IF.
2288              *
2289              ********************************************************
```

```
2290
2291 E26AC 0762          CON(5)  =!THEN
          E
2292 E26B1 4C    *!ELSE  CON(2) #C4          ELSE
2293 E26B3 54C4          NIBASC \ELS\
          35
2294 E26B9 5C            CON(2) \E\+#80
2295
2296 E26BB 0000 *ELSE    CON(5) =DOCOL
          0
2297 E26C0 0000          CON(5)  =?COMP      check for compile mode
          0
2298 E26C5 0000          CON(5)  =TWO        2
          0
2299 E26CA 0000          CON(5)  =?PAIR      ?PAIRS
          0
2300 E26CF 0000          CON(5)  =COMPL      COMPILE
          0
2301 E26D4 0000          CON(5)  =BRNCH      BRANCH
          0
2302 E26D9 95B2          CON(5)  =HERE       leave addr for THEN
          E
2303 E26DE 0000          CON(5)  =ZERO       0
          0
2304 E26E3 0000          CON(5)  =COMMA      ALLOT SPACE FOR ADDR TO BE FILLED I
          0
2305 E26E8 9D42          CON(5)  =SWAP       swap the HERE with the addr of the
          E
2306 E26ED 0000          CON(5)  =TWO        2
          0
2307 E26F2 A762          CON(5)  =THEN       execute THEN to fill in previous ad
          E
2308 E26F7 0000          CON(5)  =TWO        leave HERE and 2 for
          0
2309 E26FC 0000          CON(5)  =SEMI       ;
          0
```



```
2310            EJECT
2311          ************************************************
2312          *
2313          * LOOP   compile time: ( addr 3 - )   immediate
2314          * compiles XLOOP
2315          *
2316          ************************************************
2317
2318 E2701 1B62          CON(5) =!ELSE
          E
2319 E2706 4C    =!LOOP  CON(2) #C4
2320 E2708 C4F4          NIBASC \LOO\
          F4
2321 E270E 0D            CON(2) \P\+#80
2322 E2710 0000 =LOOP    CON(5) =DOCOL
          0
2323 E2715 0000          CON(5)  =?COMP      ensure compile mode
          0
2324 E271A 0000          CON(5)  =THREE      3
          0
2325 E271F 0000          CON(5)  =?PAIR      ?PAIRS
          0
2326 E2724 0000          CON(5)  =COMPL      COMPILE
          0
2327 E2729 0000          CON(5)  =XLOOP      XLOOP
          0
2328 E272E 95B2          CON(5)  =HERE       HERE
          E
2329 E2733 0000          CON(5)  =MINUS      -
          0
2330 E2738 0000          CON(5)  =COMMA      ,
          0
2331 E273D 95B2          CON(5)  =HERE       PASS THIS TO SETLEV
          E
2332 E2742 C472          CON(5)  =SETLEV     HANDLE ANY LEAVES
          E
2333 E2747 0000          CON(5)  =SEMI       ;
          0
2334
2335
2336 E274C 1572 =SETLEV CON(5) $SETLV
          E
2337 E2751 143  $SETLV  A=DAT1 A            GET HERE ADDR
2338 E2754 174          D1=D1+ 5            THROW AWAY IN ANY EVENT
2339 E2757 8E00         GOSUBL =SAVEFP      SAVE FORTH ENVIRONMENT, A(A) NOT CH
          00
2340 E275D 1B48         D0=(5) =oCSP        GET STK PTR VALUE FROM WHEN  WE BEG
          BF2
2341 E2764 146          C=DAT0 A            COMPILE MODE,
2342 E2767 D7           D=C    A            SAVE IN D FOR LATER COMPARES
2343 E2769 3460         LC(5)  6            LEAVE FLAG WE WILL LOOK FOR
          000
2344 E2770 D8           B=A    A            SAVE HERE ADDR IN B
2345 E2772 179  SETL0   D1=D1+ 10           MAKE SURE THERE ARE 2 ITEMS ON
2346 E2775 137          CD1EX               STK SO THAT WE DON'T GO BELOW
```

```
2347 E2778 8B7          ?C>D   A          CONTROLLED VALUE
2348 E277B 01           GOYES  NONE       NONE TO DO
2349 E277D 137          CD1EX
2350 E2780 1C9          D1=D1- 10         RESTORE STK PTR
2351
2352 E2783 143          A=DAT1 A          READ FLAG OF CONDITIONAL PAIR
2353 E2786 8A2          ?A=C   A          MATCH WITH LEAVE?
2354 E2789 80           GOYES  SETL1      YES!!
2355 E278B 8C00 NONE    GOLONG =GETFP     NO MORE LEAVES TO HANDLE
          00
2356           *                          RESTORE FORTH ENVIRONMENT
2357
2358 E2791 174  SETL1   D1=D1+ 5          THROW 6 AWAY
2359 E2794 143          A=DAT1 A          READ LEAVE ADDR
2360 E2797 130          D0=A
2361 E279A EC           A=B-A  A          HERE-LEAVE ADDR
2362 E279C 140          DAT0=A A          WRITE BRANCH OFFSET INTO LEAVE
2363 E279F 174          D1=D1+ 5          THROW AWAY LEAVE ADDR
2364 E27A2 1BDF         D0=(5) =DSTKAD    SET DATA STK PTR TO REFLECT
          AF2
2365 E27A9 137          CD1EX             LOSS OF LEAVE PAIR
2366 E27AC 144          DAT0=C A
2367 E27AF 137          CD1EX
2368 E27B2 6FBF         GOTO   SETL0      TRY FOR ANOTHER LEAVE
```



```
2369                    EJECT
2370           ************************************************
2371           *
2372           * ROLL  ( n --- ) Rotate items on stack to bring
2373           *       nth item to top of stack.
2374           *
2375           ************************************************
2376
2377 E27B6 6072         CON(5)  =!LOOP
          E
2378 E27BB 48   =!ROLL  CON(2)  #84           ROLL
2379 E27BD 25F4         NIBASC  \ROL\
          C4
2380 E27C3 CC           CON(2)  \L\+#80
2381 E27C5 0000 =ROLL   CON(5)  =DOCOL        :
          0
2382 E27CA B591         CON(5)   =DUP         if n is less than 1
          E
2383 E27CF 0000         CON(5)   =ONE         abort with less than 1
          0
2384 E27D4 0000         CON(5)   =LT          parameter message
          0
2385 E27D9 0000         CON(5)   =ABORTW
          0
2386 E27DE 60           CON(2)  =eLT1
2387 E27E0 0000         CON(5)   =ONE+        1+  add 1 to n
          0
2388 E27E5 B591         CON(5)   =DUP         DUP
          E
2389 E27EA 1052         CON(5)   =PICK        PICK
          E
2390 E27EF 9D42         CON(5)   =SWAP        SWAP
          E
2391 E27F4 45C1         CON(5)   =LIT
          E
2392 E27F9 5000         CON(5)   5            5
          0
2393 E27FE 0000         CON(5)   =MULT        *
          0
2394 E2803 5881         CON(5)   =SP@         SP@
          E
2395 E2808 0000         CON(5)   =ADD         +
          0
2396 E280D B591 ROLL1   CON(5)   =DUP         BEGIN DUP
          E
2397 E2812 0000         CON(5)   =FIVE-       5-
          0
2398 E2817 0000         CON(5)   =AT          @
          0
2399 E281C 3352         CON(5)   =OVER        OVER
          E
2400 E2821 0000         CON(5)   =STORE       !
          0
2401 E2826 0000         CON(5)   =FIVE-       5-
          0
```

```
2402 E282B 5881E        CON(5)   =SP@        SP@
2403 E2830 3352E        CON(5)   =OVER       OVER
2404 E2835 00000        CON(5)   =U<         U<
2405 E283A 00000        CON(5)   =ZEQ        0=
2406 E283F 00000        CON(5)   =ZBRNH      UNTIL
2407 E2844 9CFFF        CON(5)   (ROLL1)-*
2408 E2849 00000        CON(5)   =2DROP      2DROP  (OR DDROP)
2409 E284E 00000        CON(5)   =SEMI       ;
```



```
2410                    EJECT
2411                ****************************************************************
2412                *
2413                * FIND  addr1 --- addr2 n      Addr1 is the address of a counted
2414                *       string.  The string contains a word name to be located in
2415                *       the currently active search order.  If the word is found,
2416                *       addr2 is the compilation address and n is set to  1 if
2417                *       the word has the immediate attribute, or -1 if the word
2418                *       is non-immediate.  If the word is not found then
2419                *       add2=addr1 and n = 0.
2420                *
2421                ****************************************************************
2422
2423 E2853 BB72E            CON(5)   =!ROLL
2424 E2858 48       =!FIND  CON(2)   #84
2425 E285A 6494E4           NIBASC   \FIN\
2426 E2860 4C               CON(2)   \D\+#80
2427
2428 E2862 00000  =FIND     CON(5)   =DOCOL      :
2429 E2867 B591E            CON(5)   =DUP        (* ADDR1  ADDR1 *)
2430 E286C 00000            CON(5)   =CONTX      CONTEXT
2431 E2871 00000            CON(5)   =AT         @
2432 E2876 00000            CON(5)   =AT         @  (* ADDR1 ADDR1  SEARCH-ADDR *)
2433 E287B 00000            CON(5)   =FIND>      FIND> RETURNS TWO POSSIBLE SETS OF
2434                    *                        ADDR1  CFA-OF-WORD  LEN-OF-WORD  TR
2435                    *                        ADDR1  FALSE
2436 E2880 00000            CON(5)   =ZBRNH      IF  WORD WAS FOUND
2437 E2885 64000            CON(5)   (FND0)-*    DO THE FOLLOWING:
2438                    *                        len-of-word returned by FIND> is th
2439                    *                        raw length from the word's NFA so i
2440                    *                        includes the information we are loo
2441                    *                        for, so . . .
2442
2443 E288A 45C1E            CON(5)   =LIT
2444 E288F 04000            CON(5)   #40         40 (HEX)
2445 E2894 7A91E            CON(5)   =AND        SEE IF IMMEDIATE BIT IS SET
2446 E2899 00000            CON(5)   =ZBRNH      IF WORD WAS IMMEDIATE,
2447 E289E 41000            CON(5)   (FND01)-*
2448 E28A3 00000            CON(5)   =ONE        PUT 1 ON STACK
```

```
2449 E28A8 0000          CON(5)  =BRNCH
           0
2450 E28AD F000          CON(5)  (FND02)-*
           0
2451 E28B2 45C1 FND01    CON(5)  =LIT          ELSE PUT -1 ON STACK
           E
2452 E28B7 FFFF          CON(5)  (0-1)
           F
2453 E28BC 8791 FND02    CON(5)  =ROT          (*CFA -1/1  ADDR1*)
           E
2454 E28C1 0C42          CON(5)  =DROP         (*CFA  -1/1 *)
           E
2455 E28C6 0000          CON(5)  =SEMI         CHEAT AND END HERE
           0
2456
2457 E28CB 0000 FND0     CON(5)  =ZERO         (* ADDR1  0 *)
           0
2458 E28D0 0000 FND03    CON(5)  =SEMI         ;
           0
```



```
2459                     EJECT
2460              ***********************************************************
2461              *
2462              * QUIT  ( --- ) Clear the return stack, set execution mode,
2463              *       and return control to the terminal. No message is
2464              *       given.
2465              *
2466              ***********************************************************
2467              *
2468
2469 E28D5 8582          CON(5)  =!FIND
           E
2470 E28DA 48    =!QUIT  CON(2)  #84             QUIT
2471 E28DC 1555          NIBASC  \QUI\
           94
2472 E28E2 4D            CON(2)  \T\+#80
2473
2474 E28E4 0000 =QUIT     CON(5)  =DOCOL          :
           0
2475 E28E9 0000          CON(5)   =ZERO          set BLK to 0 signalling
           0
2476 E28EE DCA1          CON(5)   =BLK           execution from terminal
           E
2477 E28F3 0000          CON(5)   =STORE         instead of a screen
           0
2478 E28F8 0000          CON(5)   =LBRAC         [ set execution mode
           0
2479 E28FD 0792 QUIT00   CON(5)   =CR?           CR IF NEEDED
           E
2480              *                            peculiarity of mainframe--one cr doe
2481              *                            clear LCD (with OK in it), but 2nd c
2482
2483 E2902 5391 QUIT01   CON(5)   =RP!           reset return stack
           E
2484 E2907 0000          CON(5)   =QUERY         get user input
           0
2485 E290C 0000          CON(5)   =INTRP         interpret user input
           0
2486 E2911 0000          CON(5)   =?STK          ?STACK  CHECK STACK DEPTH!
           0
2487 E2916 0000          CON(5)   =OKFLG
           0
2488 E291B 0000          CON(5)   =AT
           0
2489 E2920 0000          CON(5)   =STATE
           0
2490 E2925 0000          CON(5)   =AT
           0
2491 E292A 0000          CON(5)   =OR
           0
2492              *
2493              * IF THE CONTENTS OF BOTH STATE AND THE OKFLAG
2494              * ARE ZERO THEN WE CAN GIVE THE OK MESSAGE
2495              *
2496 E292F 0000          CON(5)   =ZEQ           0=
```

```
          0
2497 E2934 0000          CON(5)   =ZBRNH          0BRANCH
          0
2498 E2939 D200          CON(5)  (QUIT0)-*
          0
2499 E293E 0000 QUITX    CON(5)   =PDOTQ          ."
          0
2500 E2943 60            CON(2)  6
2501 E2945 02F4          NIBASC  \ OK { \
          B402
          B702
2502 E2951 0000          CON(5)   =DEPTH          DEPTH
          0
2503 E2956 0000          CON(5)   =DOT            .
          0
2504 E295B 0000          CON(5)   =PDOTQ          ."
          0
2505 E2960 20            CON(2)  2
2506 E2962 D702          NIBASC  \} \
2507 E2966 0000 QUIT0    CON(5)   =BRNCH          BRANCH
          0
2508 E296B 29FF          CON(5)  (QUIT00)-*
          F
2509
2510             * DO CR IF NEEDED
2511 E2970 5792 =CR?     CON(5)  =$CR?
          E
2512 E2975 8E00 =$CR?    GOSUBL  =SAVEFP
          00
2513 E297B 1B00          D0=(5)  (=DSPSTA)+3   POINT TO STATUS BITS
          000
2514 E2982 1563          C=DAT0  X             READ STATUS BITS TO C(X)
2515 E2986 0B            CSTEX                 PUT INFO INTO STATUS BITS
2516 E2988 870           ?ST=1   =Clear       CR ALREADY SENT?
2517 E298B 90            GOYES   CRDONE
2518 E298D 8F00          GOSBVL  =FINLIN
          000
2519 E2994 8E00 CRDONE   GOSUBL  =GETFP
          00
2520 E299A 03            RTNCC
2521
2522             * ROUTINE TO WAKE UP WITH OK PROMPT;(ALTERNATIVE
2523             * OUTER LOOP, used when waking up from deep sleep -- want
2524             * to see OK without resetting stack
2525 E299C 0000 =WAKE     CON(5)  =DOCOL
          0
2526 E29A1 0000          CON(5)  =BRNCH
          0
2527 E29A6 89FF          CON(5)  (QUITX)-*
          F
```



```
2528             EJECT
2529          ******************************************************
2530          *
2531          * WORD (char --- addr) Receive chars from input stream
2532          *       until non-zero delimiting char is encountered or
2533          *       the input stream exhausted, ignoring leading
2534          *       delimiters.  The chars are moved to HERE in the
2535          *       form:
2536          *       (HERE) <char cnt> <chars>
2537          *       If input stream is exhausted as WORD is called,
2538          *       then a zero length is left at HERE
2539          *
2540          ******************************************************
2541
2542 E29AB AD82          CON(5)  =!QUIT
          E
2543 E29B0 48    =!WORD  CON(2)  #84             WORD
2544 E29B2 75F4          NIBASC  \WOR\
          25
2545 E29B8 4C            CON(2)  \D\+#80
2546
2547 E29BA 0000 =WORD    CON(5)  =DOCOL          :
          0
2548 E29BF B591 WRD0     CON(5)   =DUP           DUP CHAR TO MATCH
          E
2549 E29C4 0000          CON(5)   =STRM          get addr to find next word at
          0
2550 E29C9 9D42          CON(5)   =SWAP          char addr char
          E
2551 E29CE 0000          CON(5)   =ENCL          returns addr and 3 offsets
          0
2552 E29D3 5552          CON(5)   =2DUP          dup 2 flags for null test
          E
2553 E29D8 0000          CON(5)   =GT            >
          0
2554 E29DD 0000          CON(5)   =ZBRNH         IF null then
          0
2555 E29E2 8200          CON(5)  (WD)-*
          0
2556 E29E7 0000          CON(5)   =2DROP         clear items off stack
          0
2557 E29EC 0000          CON(5)   =2DROP
          0
2558 E29F1 0000          CON(5)   =ZERO          store a 0 (meaning null word)
          0
2559 E29F6 95B2          CON(5)   =HERE          and HERE
          E
2560 E29FB 0000          CON(5)   =STORE
          0
2561 E2A00 0000          CON(5)   =BRNCH         ELSE
          0
2562 E2A05 0500          CON(5)  (WD0)-*
          0
2563 E2A0A AEA1 WD       CON(5)   =IN            >IN  use n3 from enclose
          E
```

```
2564 E2A0F 0000          CON(5)   =PSTOR          +!   to update pointer into TIB
             0
2565 E2A14 3352          CON(5)   =OVER                next get the # nibs involve
             E
2566 E2A19 0000          CON(5)   =MINUS           -    this word
             0
2567 E2A1E B591          CON(5)   =DUP             DUP  put a copy on the rtn stack
             E
2568 E2A23 0000          CON(5)   =>R              >R
             0
2569 E2A28 0000          CON(5)   =TWO/            2/   turn nibbles to bytes
             0
2570 E2A2D 95B2          CON(5)   =HERE            HERE store byte count of word at
             E
2571 E2A32 0000          CON(5)   =C!              C!
             0
2572 E2A37 0000          CON(5)   =ADD             +    compute source
             0
2573 E2A3C 95B2          CON(5)   =HERE            HERE
             E
2574 E2A41 0000          CON(5)   =TWO+            2+   destination
             0
2575 E2A46 0000          CON(5)   =R>              R>   # nibs to move
             0
2576 E2A4B 0000          CON(5)   =TWO+            TWO+
             0
2577 E2A50 0000          CON(5)   =NMOVE           NMOVE  move them
             0
2578 E2A55 95B2 WD0      CON(5)   =HERE            HERE   word is now at HERE
             E
2579
2580                *
2581                * AT THIS POINT WE HAVE AN ADDR AND A CHAR ON STACK
2582                *
2583
2584 E2A5A B591            CON(5)   =DUP             DUP ADDR
             E
2585 E2A5F 0000          CON(5)   =TWO+            POINT TO 1ST CHAR IN WORD
             0
2586 E2A64 0000          CON(5)   =CAT             C@ GET 1ST CHAR
             0
2587 E2A69 0000          CON(5)   =ZEQ             0= TEST FOR NULL
             0
2588 E2A6E 0000          CON(5)   =ZBRNH           BRANCH IF NOT NULL
             0
2589 E2A73 A500          CON(5)   (WD00)-*
             0
2590 E2A78 DCA1          CON(5)   =BLK             BLK
             E
2591 E2A7D 0000          CON(5)   =AT              @   SEE IF WE'RE LOADING
             0
2592 E2A82 0000          CON(5)   =ZBRNH           BRANCH IF EXECUTING
             0
2593 E2A87 6400          CON(5)   (WD00)-*
             0
```



```
2594 E2A8C 33F1          CON(5)   =EOF             WE'RE LOADING, IS IT EOF?
             E
2595 E2A91 0000          CON(5)   =ZEQ             0=
             0
2596 E2A96 0000          CON(5)   =ZBRNH           BRANCH IF EOF
             0
2597 E2A9B 2300          CON(5)   (WD00)-*
             0
2598 E2AA0 0000          CON(5)   =ONE             1
             0
2599 E2AA5 DCA1          CON(5)   =BLK             BLK    INCR LINE NUMBER
             E
2600 E2AAA 0000          CON(5)   =PSTOR           +!
             0
2601 E2AAF 0000          CON(5)   =ZERO            0
             0
2602 E2AB4 AEA1          CON(5)   =IN              >IN   RESET TIB POINTER
             E
2603 E2AB9 0000          CON(5)   =STORE           !
             0
2604 E2ABE 0C42          CON(5)   =DROP            DROP  ADDR
             E
2605 E2AC3 0000          CON(5)   =BRNCH           GO AGAIN  LOOK FOR WORD
             0
2606 E2AC8 7FEF          CON(5)   (WRD0)-*         IN NEXT LINE OF FILE
             F
2607 E2ACD 9D42 WD00     CON(5)   =SWAP            CHAR TO TOP OF STACK
             E
2608 E2AD2 0C42          CON(5)   =DROP            DROP CHAR, LEAVE ADDR
             E
2609 E2AD7 0000          CON(5)   =SEMI              ;
             0
```

```
                        EJECT
                *************************************************************
                *
                * EMIT (c --- ) Transmit a char to the current output device.
                *
                *************************************************************
2615 E2ADC 0B92         CON(5)  =!WORD
           E
2617 E2AE1 48   =!EMIT  CON(2)  #84             EMIT
2618 E2AE3 54D4         NIBASC  \EMI\
           94
2619 E2AE9 4D           CON(2)  \T\+#80
2620 E2AEB 0FA2 =EMIT   CON(5)  =$EMIT
           E
2621
2622 E2AF0 8F00 =$EMIT  GOSBVL  =SAVEFP         SAVE FORTH PTRS
           000
2623 E2AF7 D0           A=0     A
2624 E2AF9 E4           A=A+1   A               A=1=# OF CHARS
2625                * D1 ALREADY POINTS TO THE "OUTPUT BUFFER", I.E. THE CHAR
2626 E2AFE 101          R1=A
2627 E2AFE 137          CD1EX
2628 E2B01 10B          R3=C                    SAVE D1 FROM CKINF
2629                *
2630 E2B04 7C00         GOSUB   DSPSUB
2631                *
2632 E2B08 8F00         GOSBVL  =GETFP          RESTORE FORTH POINTERS
           000
2633 E2B0F 174          D1=D1+  5               RETURN NOTHING
2634 E2B12 03           RTNCC
2635
2636 E2B14 8F00 DSPSUB  GOSBVL  =CKINF-         SET UP FOR SENDWD
           000
2637 E2B1B 11B          C=R3
2638 E2B1E 135          D1=C
2639 E2B21 111          A=R1
2640 E2B24 854          ST=1    4
2641 E2B27 8F00         GOSBVL  =SENDWD
           000
2642                *
2643                * WHEN THERE HAS BEEN NO POLL HANDLED IN CKINF- OR SENDWD,
2644                * FOLLOWING AMOUNTS TO A GOVLNG =DOSCRL (SCROLL)
2645                *
2646 E2B2E 1F00          D1=(5)  (=STMTR0)+1     POINT TO "PART2" OF DSP STUFF
           000
2647 E2B35 147          C=DAT1  A               GET ADDR OF IT
2648 E2B38 135          D1=C
2649 E2B3B 1C4          D1=D1-  5          "PART 3" OFFSET IS IN FRONT OF IT
2650 E2B3E 147          C=DAT1  A          GET OFFSET
2651 E2B41 133          AD1EX
2652 E2B44 C2           C=C+A   A          ADD ADDR
2653 E2B46 06           RSTK=C             PUT ON RSTK
2654 E2B48 03           RTNCC              GO THERE
```



```
                        EJECT
                *************************************************************
                *
                * HERE  ( --- addr) Returns addr of next available dict.
                *       location.
                *
                *************************************************************
2662
2663 E2B4A 1EA2         CON(5)  =!EMIT
           E
2664 E2B4F 48   =!HERE  CON(2)  #84
2665 E2B51 8454         NIBASC  \HER\
           25
2666 E2B57 5C           CON(2)  \E\+#80
2667
2668 E2B59 0000 =HERE   CON(5)  =DOCOL          :
           0
2669 E2B5E 0000         CON(5)  =DP             DP
           0
2670 E2B63 0000         CON(5)  =AT             @
           0
2671 E2B68 0000         CON(5)  =SEMI           ;
           0
```

```
2672                    EJECT
2673                ********************************************************
2674                *
2675                * /MOD  (n1 n2 --- n3 n4) Divide n1 by n2 leave remainder
2676                *       n3 and quotient n4
2677                *
2678                ********************************************************
2679
2680 E2B6D F4B2            CON(5)  =!HERE
           E
2681 E2B72 48     =!/MOD    CON(2)  #84             /MOD
2682 E2B74 F2D4            NIBASC  \/MO\
           F4
2683 E2B7A 4C              CON(2)  \D\+#80
2684
2685 E2B7C 0000 =/MOD      CON(5)  =DOCOL          :
           0
2686 E2B81 0000            CON(5)   =>R            n1 :: n2
           0
2687 E2B86 5F52            CON(5)   =S->D          d1 :: n2
           E
2688 E2B8B 0000            CON(5)   =R>            d1 n2
           0
2689 E2B90 0000            CON(5)   =M/            rem quot
           0
2690 E2B95 0000            CON(5)   =SEMI           ;
           0
2691
2692                ****************************************************************
2693                *
2694                * HOLD  (char --- )  Insert char into pictured numeric output str
2695                *       Use between <# and #>
2696                *
2697                ****************************************************************
2698
2699 E2B9A 27B2            CON(5)  =!/MOD
           E
2700 E2B9F 48     =!HOLD    CON(2)  #84             HOLD
2701 E2BA1 84F4            NIBASC  \HOL\
           C4
2702 E2BA7 4C              CON(2)  \D\+#80
2703
2704 E2BA9 0000 =HOLD      CON(5)  =DOCOL          : HOLD
           0
2705 E2BAE 45C1            CON(5)   =LIT           decrement addr by 2 inorder to
           E
2706 E2BB3 EFFF            CON(5)  (0-2)           to insert this char
           F
2707 E2BB8 A2B1            CON(5)   =HLD           HLD
           E
2708 E2BBD 0000            CON(5)   =PSTOR         +!
           0
2709 E2BC2 A2B1            CON(5)   =HLD           HLD   store char at this new
           E
2710 E2BC7 0000            CON(5)   =AT            @     addr
```



```
           0
2711 E2BCC 0000            CON(5)   =C!            C!
           0
2712 E2BD1 0000            CON(5)   =SEMI          ;
           0
```

```
2713                    EJECT
2714                ********************************************
2715                *
2716                * ?DUP   (n --- [n] n)  dup if non-zero
2717                *
2718                ********************************************
2719
2720 E2BD6 F9B2           CON(5)  =!HOLD
         E
2721 E2BDB 48     =!?DUP  CON(2)  #84             ?DUP
2722 E2BDD F344           NIBASC  \?DU\
         55
2723 E2BE3 0D           CON(2)  \P\+#80
2724 E2BE5 AEB2 =?DUP   CON(5)  $?DUP
         E
2725 E2BEA 143  $?DUP    A=DAT1  A               READ n
2726 E2BED 8A8           ?A=0    A               0?
2727 E2BF0 00            RTNYES                  RETURN IF YES (CARRY IS SET)
2728 E2BF2 1C4           D1=D1-  5               n <> 0 SO DUP IT
2729 E2BF5 141           DAT1=A  A
2730 E2BF8 03            RTNCC
2731
2732                ****************************************************************
2733                *
2734                * SIGN ( n - ) Insert the ASCII "-" (minus sign) into the picture
2735                * numeric output string, if n is negative.
2736                *
2737                ****************************************************************
2738
2739 E2BFA BDB2           CON(5)  !?DUP
         E
2740 E2BFF 48     =!SIGN  CON(2)  #84             SIGN
2741 E2C01 3594          NIBASC  \SIG\
         74
2742 E2C07 EC            CON(2)  \N\+#80
2743
2744 E2C09 0000 =SIGN    CON(5) =DOCOL
         0
2745 E2C0E 0000          CON(5) =LTZ            if n < 0 then
         0
2746 E2C13 0000          CON(5) =ZBRNH          store a minus sign
         0
2747 E2C18 4100          CON(5) (SIGN00)-*      in the HOLD area
         0
2748 E2C1D 45C1          CON(5) =LIT            (area between HERE and
         E
2749 E2C22 D200          CON(5) \-\             PAD)
         0
2750 E2C27 9AB2          CON(5) =HOLD
         E
2751 E2C2C 0000 SIGN00   CON(5) =SEMI
         0
```



```
2752                    EJECT
2753                ********************************************************
2754                *
2755                * ?CSP  Issue error message if stack position differs from
2756                *       value saved in CSP.
2757                *
2758                ********************************************************
2759
2760 E2C31 0000 =QCSP   CON(5)  =DOCOL          :
         0
2761 E2C36 5881         CON(5)  =SP@           get current top of stack
         E
2762 E2C3B A1B1         CON(5)  =CSP           get stack level stored in CSP
         E
2763 E2C40 0000         CON(5)  =AT            @
         0
2764 E2C45 0000         CON(5)  =MINUS         - error if diff # 0
         0
2765 E2C4A 0000         CON(5)  =ABRT"X        ABORT"
         0
2766 E2C4F 70           CON(2)  =eNDEF
2767                *    NIBASC  \ definit\
2768                *    NIBASC  \ion not \
2769                *    NIBASC  \finished\
2770 E2C51 0000         CON(5)  =SEMI          ;
         0
```

```
2771                    EJECT
2772                ******************************************************
2773                *
2774                * CASE
2775                *  Implements words to allow, e.g.:
2776                *                  CASE n1 OF action1 ENDOF action2
2777                *                       n2 OF action3  ENDOF action4
2778                *                       ...
2779                *                  ENDCASE
2780                *
2781                * The word CASE itself does nothing except make a syntactic
2782                * requirement, i.e. OF won't work unless there has been a CASE.
2783                * I guess this has an aesthetic appeal as providing symmetry
2784                * for the ENDCASE.
2785                ******************************************************
2786
2787 E2C56 FFB2             CON(5)  =!SIGN
       E
2788 E2C5B 4C     =!CASE    CON(2)  #C4
2789 E2C5D 3414             NIBASC  \CAS\
       35
2790 E2C63 5C               CON(2)  \E\+#80
2791
2792 E2C65 0000 =CASE      CON(5) =DOCOL       :
       0
2793 E2C6A 0000            CON(5)   =?COMP      ?COMP
       0
2794 E2C6F 45C1            CON(5)   =LIT            JUST LEAVE 8 FOR
       E
2795 E2C74 8000            CON(5)  8                SYNTAX CHECK BY OF
       0
2796 E2C79 B591             CON(5)  =DUP            2 8'S SO EVERYTHING
       E
2797 E2C7E 0000            CON(5)   =SEMI           IS IN PAIRS AT COMPILE TIME
       0
```



```
2798                    EJECT
2799                ******************************************************
2800                *
2801                * TYPE  (addr  n --- )  transmit n chars to output device.
2802                *       No action if n < = 0
2803                *
2804                ******************************************************
2805 E2C83 B5C2             CON(5)  =!CASE
       E
2806 E2C88 48     =!TYPE    CON(2)  #84              TYPE
2807 E2C8A 4595             NIBASC  \TYP\
       05
2808 E2C90 5C               CON(2)  \E\+#80
2809 E2C92 79C2 =TYPE      CON(5)  =$TYPE
       E
2810
2811 E2C97 143  =$TYPE     A=DAT1  A                POP N
2812 E2C9A 8A8             ?A=0    A                BAD?
2813 E2C9D D2              GOYES   TPO              YES
2814 E2C9F C4              A=A+A   A                NEQ #?
2815 E2CA1 482             GOC     TPO              YES
2816 E2CA4 143             A=DAT1  A                N
2817 E2CA7 174             D1=D1+  5                PT TO ADDR
2818 E2CAA 147             C=DAT1  A                C=ADDR
2819 E2CAD 174             D1=D1+  5                RETURN NOTHING ON DATA STK
2820 E2CB0 10B             R3=C                     SAVE ADDR
2821 E2CB3 8F00            GOSBVL  =SAVEFP          SAVE FORTH POINTERS, A UNTOUCHED
       000
2822 E2CBA 101             R1=A                     SAVE SIZE
2823 E2CBD 735E            GOSUB   DSPSUB
2824                *
2825 E2CC1 8F00            GOSBVL  =GETFP           RECOVER FORTH POINTERS
       000
2826 E2CC8 03              RTNCC
2827 E2CCA 179  TPO        D1=D1+  10               RETURN NOTHING
2828 E2CCD 03              RTNCC
```

```
2829                    EJECT
2830                ***********************************
2831                *
2832                * ADVANCE IN PTR ONE CHARACTER
2833                *
2834                ***********************************
2835
2836 E2CCF 0000 =PTI+  CON(5)  =DOCOL       :
             0
2837 E2CD4 0000        CON(5)  =TWO         2
             0
2838 E2CD9 AEA1        CON(5)  =IN          >IN
             E
2839 E2CDE 0000        CON(5)  =PSTOR       +!
             0
2840 E2CE3 0000        CON(5)  =SEMI        ;
             0
```



```
2841                    EJECT
2842                ***************************************************************
2843                *
2844                * +BUF (addr1 --- addr2  f)  This routine advances the disc
2845                *      buffer address, addr1 to address of next buffer.  f is
2846                *      false when addr2 is buffer presently pointed to by PREV
2847                *
2848                ***************************************************************
2849
2850 E2CE8 88C2             CON(5)  =!TYPE
             E
2851 E2CED 48    =!PBUF    CON(2) #84            +BUF
2852 E2CEF B224            NIBASC \+BU\
             55
2853 E2CF5 6C              CON(2) \F\+#80
2854
2855 E2CF7 0000 =PBUF     CON(5)  =DOCOL        :
             0
2856 E2CFC 0000           CON(5)    =BSIZE      BFSIZE   get buffer size in NIBS
             0
2857 E2D01 0000           CON(5)    =AT         @
             0
2858 E2D06 0000           CON(5)    =ADD        +     add it to current buffer ad
             0
2859 E2D0B B591           CON(5)    =DUP        DUP   duplicate it
             E
2860 E2D10 0000           CON(5)    =LIMIT      LIMIT get addr of 1st byte beyond
             0
2861 E2D15 0000           CON(5)    =AT         @      last buffer
             0
2862 E2D1A 0000           CON(5)    =EQUAL      =     compare the 2,
             0
2863 E2D1F 0000           CON(5)    =ZBRNH      IF    if they're the same
             0
2864 E2D24 4100           CON(5)   (PBUF0)-*          then we've gone too far
             0
2865 E2D29 0C42           CON(5)    =DROP       DROP  drop this addr
             E
2866 E2D2E 0000           CON(5)    =FIRST      FIRST    get addr of 1st buffer
             0
2867 E2D33 0000           CON(5)    =AT         @
             0
2868 E2D38 B591 PBUF0     CON(5)    =DUP        THEN DUP   duplicate it
             E
2869 E2D3D C362           CON(5)    =PREV       PREV     fetch addr of PREV
             E
2870 E2D42 0000           CON(5)    =AT         @      buffer
             0
2871 E2D47 0000           CON(5)    =<>         if same return 0 else return -1
             0
2872 E2D4C 0000           CON(5)    =SEMI       ;     status!
             0
```

```
2873                    EJECT
2874                ********************************************************
2875                *
2876                * SUB$                          ( str pos1 pos2 -- substr )
2877                * Return temporary string which is a copy of
2878                * the given string from the pos1 character to the
2879                * pos2 character.
2880                *
2881                ********************************************************
2882
2883 E2D51 DEC2     =!PBUF  CON(5)  =!PBUF
       E
2884 E2D56 48       =!SUB$  CON(2)  #84
2885 E2D58 3555     SUB$    NIBASC  \SUB\
       24
2886 E2D5E 4A               CON(2)  \$\+#80
2887
2888 E2D60 0000     =SUB$   CON(5)  =DOCOL
       0
2889 E2D65 0000             CON(5)  =DEPTH
       0
2890 E2D6A 0000             CON(5)  =THREE
       0
2891 E2D6F 0000             CON(5)  =GT             DEPTH MUST BE >3
       0
2892 E2D74 0000             CON(5)  =ZEQ
       0
2893 E2D79 0000             CON(5)  =ZBRNH
       0
2894 E2D7E 1100             CON(5)  (SUB$1)-*
       0
2895 E2D83 0000             CON(5)  =ONE
       0
2896 E2D88 0000             CON(5)  =ABORTW
       0
2897 E2D8D E2               CON(2)  =eBADPM     "bad parameters"
2898                ** * * * * * * * * * * * * * * * * * * * *
2899                * ADR LEN P1 P2
2900 E2D8F 8791     SUB$1   CON(5)  =ROT        ROT
       E
2901                *
2902                * ADR P1 P2 LEN
2903                *
2904                * NOW MAKE LEN MOD 256 -- THIS PROTECTS AGAINST
2905                * HORRIBLE RESULTS WHICH CAN EASILY BE SET UP BY, E.G.,
2906                * TYPING IN "5 STR END$" WHEN YOU MEANT "STR 5 END$"
2907 E2D94 45C1             CON(5)  =LIT
       E
2908 E2D99 FF00             CON(5)  255
       0
2909 E2D9E 7A91             CON(5)  =AND
       E
2910                * ADR P1 P2 LEN
2911 E2DA3 21D1             CON(5)  =MIN        MIN
       E
```



```
2912                * use pos2 or len of str, whichever is less
2913 E2DA8 0000             CON(5)  =>R
       0
2914                * ADR P1        --      MIN(P2,LEN)
2915 E2DAD 0000             CON(5)  =ONE
       0
2916 E2DB2 74D1             CON(5)  =MAX
       E
2917                * use p1 or 1, whichever is greater
2918 E2DB7 0000             CON(5)  =R>
       0
2919                * ADR P1 P2  (GOOD P1 & P2)
2920 E2DBC 5552             CON(5)  =2DUP       2DUP
       E
2921 E2DC1 9D42             CON(5)  =SWAP       SWAP
       E
2922 E2DC6 0000             CON(5)  =LT         <
       0
2923                * compare pos1 and last pos
2924 E2DCB 0000             CON(5)  =ZEQ        0=
       0
2925                * if pos1 > lastpos then return null string
2926 E2DD0 0000             CON(5)  =ZBRNH      IF
       0
2927 E2DD5 B900             CON(5)  (SUB01)-*
       0
2928 E2DDA 9D42             CON(5)  =SWAP       adr p2 p1
       E
2929 E2DDF B591             CON(5)  =DUP        adr p2 p1 p1
       E
2930 E2DE4 8791             CON(5)  =ROT        adr p1 p1 p2
       E
2931 E2DE9 9D42             CON(5)  =SWAP       adr p1 p2 p1
       E
2932 E2DEE 0000             CON(5)  =MINUS      adr p1 (p2-p1)
       0
2933 E2DF3 0000             CON(5)  =ONE+       1+
       0
2934                * got # of chars in substring
2935 E2DF8 8791             CON(5)  =ROT        p1 count adr
       E
2936 E2DFD 8791             CON(5)  =ROT        count adr p1
       E
2937 E2E02 0000             CON(5)  =ONE-       count adr p1-1
       0
2938 E2E07 0000             CON(5)  =TWO*       2*
       0
2939                * chars-->nibs
2940 E2E0C 0000             CON(5)  =ADD        count pos
       0
2941                * position of first character of substring
2942 E2E11 3352             CON(5)  =OVER       count pos count
       E
2943 E2E16 63C1             CON(5)  =PAD        count pos count pad
       E
```

```
2944 E2E1B 5552          CON(5)   =2DUP       c pos c pad c pad
         E
2945 E2E20 3352          CON(5)   =OVER       c pos c pad c pad c
         E
2946 E2E25 45C1          CON(5)   =LIT
         E
2947 E2E2A 0500          CON(5)   80            MAXLEN=80
         0
2948 E2E2F 74D1          CON(5)   =MAX        c pos c pad c pad max(c,80)
         E
2949 E2E34 63C1          CON(5)   =PAD
         E
2950 E2E39 0000          CON(5)   =C!         store max(len,80) at pad
         0
2951 E2E3E 0000          CON(5)   =TWO+       c pos c pad c pad+2
         0
2952 E2E43 0000          CON(5)   =C!         store char-count at pad+2
         0
2953             * stored maxlen & len at PAD
2954 E2E48 0000          CON(5)   =FOUR+      c pos c pad+4
         0
2955 E2E4D 9D42          CON(5)   =SWAP       c pos pad+4 c
         E
2956 E2E52 0000          CON(5)   =CMOVE      move c chars from pos to pad+4
         0
2957 E2E57 63C1          CON(5)   =PAD        c pad
         E
2958 E2E5C 0000          CON(5)   =FOUR+      c pad+4
         0
2959 E2E61 9D42          CON(5)   =SWAP       pad+4 c  = the string
         E
2960 E2E66 0000          CON(5)   =BRNCH      ELSE
         0
2961 E2E6B 4100          CON(5)  (SUB02)-*
         0
2962             * leave null string
2963 E2E70 0C42 SUB01    CON(5)   =DROP       DROP
         E
2964 E2E75 0000          CON(5)   =2DROP      2DROP
         0
2965 E2E7A 0000          CON(5)   =NULL$      NULL$
         0
2966 E2E7F 0000 SUB02    CON(5)   =SEMI       ;
         0
```



```
2967             EJECT
2968           ****************************************
2969           *
2970           * STR$                          ( D -- str )
2971           * Convert number to string.
2972           * ((=STR$ is also the guts of D.R ))
2973           * STR$U is the user entry point, which
2974           * calls the secondary used by D.R,et.al.
2975           * <# builds string below PAD -- this string
2976           * would get trashed by other calls to D.R,
2977           * notably the one which prints out depth in
2978           * the OK prompt. So STR$U calls STR$ and then
2979           * moves result above PAD where it is a safe
2980           * temporary string.
2981           *
2982           ****************************************
2983
2984 E2E84 65D2          CON(5)   =!SUB$
         E
2985 E2E89 48    =!STR$  CON(2)   #84           STR$
2986 E2E8B 3545          NIBASC   \STR\
         25
2987 E2E91 4A            CON(2)   \$\+#80
2988 E2E93 0000 =STR$U   CON(5)   =DOCOL
         0
2989 E2E98 1BE2          CON(5)   =STR$           make string
         E
2990 E2E9D 0000          CON(5)   =ONE
         0
2991 E2EA2 3352          CON(5)   =OVER
         E
2992             * STR 1 LEN
2993 E2EA7 06D2          CON(5)   =SUB$
         E
2994 E2EAC 0000          CON(5)   =SEMI
         0
2995
2996 E2EB1 6BE2 =STR$    CON(5)   $STR$
         E
2997
2998 E2EB6 143  $STR$    A=DAT1   A             read high nibs of D1
2999 E2EB9 101           R1=A                   save copy for neg test
3000 E2EBC C4            A=A+A    A             if neg # will set carry
3001 E2EBE 580           GONC     STR00         its positive
3002 E2EC1 8E00          GOSUBL   =$DNGAT        negate D1
         00
3003 E2EC7 20    STR00   P=       0
3004 E2EC9 3439          LC(5)    =oDP
         BF2
3005 E2ED0 136           CD0EX
3006 E2ED3 108           R0=C                   save I in R0
3007 E2ED6 142           A=DAT0   A             addr 1st free nib in dictionary
3008 E2ED9 34A5          LC(5)    90            PAD is 45 bytes away
         000
3009 E2EE0 CA            A=A+C    A             A(A)=addr PAD
```

```
3010 E2EE2 1B57          D0=(5)  =oBASE
           BF2
3011 E2EE9 146           C=DAT0  A
3012 E2EEC 10B           R3=C                    save (BASE) in R3
3013 E2EEF 130           D0=A                    set D0-->PAD
3014 E2EF2 102           R2=A                    save start addr--PAD addr
3015
3016 E2EF5 11B  STR01    C=R3                    C=base
3017 E2EF8 1C4           D1=D1-  5               write base to stack
3018 E2EFB 145           DAT1=C  A               D1  base  on stack
3019 E2EFE 8E00          GOSUBL  =$M/MOD         rtns Remainder, Dbl Quotient (R,
           00
3020 E2F04 AC3           D=0     S               done flag
3021 E2F07 15F9          C=DAT1  10              read DQ
3022 E2F0B 179           D1=D1+  10              point to R
3023 E2F0E 143           A=DAT1  A               read R
3024 E2F11 1C4           D1=D1-  5
3025 E2F14 15D9          DAT1=C  10              write DQ back
3026 E2F18 29            P=      9
3027 E2F1A 91E           ?C#0    WP              any quotient left?
3028 E2F1D 50            GOYES   STR02           yes
3029 E2F1F B47           D=D+1   S               no, flag it
3030 E2F22 20   STR02    P=      0
3031 E2F24 D2            C=0     A               if R > 9 we must add 7 to move
3032 E2F26 309           LC(1)   9               beyond non-displayable symbols
3033 E2F29 8BA           ?A<=C   A               R <=9 ?
3034 E2F2C 70            GOYES   STR03           if so, no problem
3035 E2F2E 307           LC(1)   7
3036 E2F31 CA            A=A+C   A
3037 E2F33 3103 STR03    LC(2)   #30             add 30 to make it ASCII digit
3038 E2F37 CA            A=A+C   A               ASCII digit
3039 E2F39 181           D0=D0-  2               decrement for char store
3040 E2F3C 148           DAT0=A  B               write char out
3041 E2F3F 94B           ?D=0    S               any more to do?
3042 E2F42 3B            GOYES   STR01           yes
3043 E2F44 111           A=R1                    recover high nibs of D1
3044 E2F47 C4            A=A+A   A               test for neg #
3045 E2F49 5C0           GONC    STR04           pos #
3046 E2F4C 181           D0=D0-  2               decrement for sign store
3047 E2F4F 31D2          LC(2)   #2D             "-"
3048 E2F53 14C           DAT0=C  B               write out neg sign
3049 E2F56 118  STR04    C=R0                    C(A)=I
3050 E2F59 136           CD0EX                   D0=I, C=end addr
3051          *                                  D1-->Q which = 0
3052 E2F5C 174           D1=D1+  5               point to lower order nibs
3053 E2F5F 145           DAT1=C  A               write out str start addr
3054 E2F62 112           A=R2                    A(A)=addr of PAD
3055 E2F65 EA            A=A-C   A               A(A)=#nibs in string
3056 E2F67 AD0           A=0     M
3057 E2F6A 81C           ASRB                    divide by 2 for bytes
3058 E2F6D 1C4           D1=D1-  5               we assume count is positive!
3059 E2F70 141           DAT1=A  A               write out byte count
3060 E2F73 03            RTNCC
```



```
3061                EJECT
3062           ********************************************
3063           *
3064           * CHR$                        ( ascii -- str )
3065           * Make a 1-char string out of an ascii value.
3066           *
3067           ********************************************
3068
3069 E2F75 98E2            CON(5)  =!STR$
           E
3070 E2F7A 48    =!CHR$    CON(2)  #84
3071 E2F7C 3484            NIBASC  \CHR\
           25
3072 E2F82 4A              CON(2)  \$\+#80
3073
3074 E2F84 0000  =CHR$     CON(5)  =DOCOL          :
           0
3075 E2F89 0000            CON(5)   =ONE           1
           0
3076 E2F8E 63C1            CON(5)   =PAD           PAD
           E
3077 E2F93 5552            CON(5)   =2DUP          1 pad 1 pad
           E
3078 E2F98 0000            CON(5)   =C!            store 1 at pad as maxlen
           0
3079 E2F9D 0000            CON(5)   =TWO+          2+
           0
3080 E2FA2 0000            CON(5)   =C!            store 1 at pad+2 as curlen
           0
3081 E2FA7 63C1            CON(5)   =PAD           PAD
           E
3082 E2FAC 0000            CON(5)   =FOUR+         4+
           0
3083 E2FB1 0000            CON(5)   =C!            store character
           0
3084 E2FB6 63C1            CON(5)   =PAD           PAD
           E
3085 E2FBB 0000            CON(5)   =FOUR+         4+
           0
3086 E2FC0 0000            CON(5)   =ONE           1
           0
3087 E2FC5 0000            CON(5)   =SEMI          ;
           0
```

```
3088                    EJECT
3089                ***********************************************
3090                *
3091                * END$                          ( str pos -- str )
3092                * Return substring from pos to end.
3093                *
3094                * Same as saying SUB$ from pos to end(=len of string).
3095                * So since we have addr len pos we just need to do
3096                * OVER to get addr len pos len and call SUB$.
3097                *
3098                ***********************************************
3099
3100 E2FCA A7F2             CON(5)  =!CHR$
           E
3101 E2FCF 48       =!END$  CON(2)  #84
3102 E2FD1 54E4             NIBASC  \END\
           44
3103 E2FD7 4A               CON(2)  \$\+#80
3104
3105 E2FD9 0000     =END$   CON(5)  =DOCOL          :
           0
3106 E2FDE 3352             CON(5)   =OVER          OVER
           E
3107 E2FE3 06D2             CON(5)   =SUB$          SUB$
           E
3108 E2FE8 0000             CON(5)   =SEMI          ;
           0
3109
3110 E2FED          =REND2  END
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER


```
=!#TIB    Abs  926353 #E2291 -  1794  1810
=!/MOD    Abs  928626 #E2B72 -  2681  2699
=!1/X     Abs  923519 #E177F -   487   500
=!10^X    Abs  926679 #E23D7 -  1929  1956
=!2DUP    Abs  927051 #E254B -  2105  2123
=!4N@     Abs  922768 #E1490 -   141   158
=!?DUP    Abs  928731 #E2BDB -  2721  2739
=!ABS     Abs  924182 #E1A16 -   778   800
=!ACOS    Abs  926536 #E2348 -  1878  1891
=!AND     Abs  924063 #E199F -   716   735
=!ASC     Abs  925755 #E203B -  1537
=!ASIN    Abs  926497 #E2321 -  1864  1877
=!ATAN    Abs  926575 #E236F -  1892  1905
=!BASE    Abs  927313 #E2651 -  2240  2260
=!BLK     Abs  924357 #E1AC5 -   871   887
=!BRACT   Abs  923014 #E1586 -   259   274
=!BYE     Abs  924479 #E1B3F -   961  1027
=!C@+     Abs  923805 #E189D -   595   621
=!CASE    Abs  928859 #E2C5B -  2788  2805
=!CHR$    Abs  929658 #E2F7A -  3070  3100
=!CHS     Abs  922891 #E150B -   193   216
=!COS     Abs  923386 #E16FA -   435   455
=!CRLF    Abs  925847 #E2097 -  1590  1612
=!D.R     Abs  924858 #E1CBA -  1139  1164
=!DABS    Abs  927150 #E25AE -  2154  2183
=!DROP    Abs  926902 #E24B6 -  2024  2040
=!DUP     Abs  923987 #E1953 -   675   692
=!ELSE    Abs  927409 #E26B1 -  2292  2318
=!EMIT    Abs  928481 #E2AE1 -  2617  2663
=!END$    Abs  929743 #E2FCF -  3101
=!ENG     Abs  923322 #E16BA -   406   420
=!EOF     Abs  925483 #E1F2B -  1421  1465
=!EXIT    Abs  926415 #E22CF -  1827  1863
=!FABS    Abs  925788 #E205C -  1561  1589
=!FILL    Abs  927084 #E256C -  2124  2153
=!FIND    Abs  927832 #E2858 -  2424  2469
=!FIX     Abs  923197 #E163D -   341   390
=!FIOI    Abs  926296 #E2258 -  1770  1793
=!GROW    Abs  926740 #E2414 -  1957  2023
=!HERE    Abs  928591 #E2B4F -  2664  2680
=!HEX     Abs  922811 #E14BB -   159   176
=!HOLD    Abs  928671 #E2B9F -  2700  2720
=!IN      Abs  924386 #E1AE2 -   888   960
=!ITOF    Abs  926190 #E21EE -  1734  1769
=!KEY     Abs  924647 #E1BE7 -  1028  1060
=!LOG     Abs  922854 #E14E6 -   177   192
=!LOOP    Abs  927494 #E2706 -  2319  2377
=!MAX     Abs  924991 #E1D3F -  1186  1420
=!MIN     Abs  924938 #E1D0A -  1165  1185
=!MOD     Abs  924825 #E1C99 -  1121  1138
=!NOT     Abs  924154 #E19FA -   762   777
=!OVER    Abs  927017 #E2529 -  2087  2104
=!PAD     Abs  924718 #E1C2E -  1061  1120
=!PBUF    Abs  929005 #E2CED -  2851  2883
=!PICK    Abs  926967 #E24F7 -  2062  2086
```

```
=!POS    Abs  925603 #E1FA3 -  1466  1514
=!PREV   Abs  927282 #E2632 -  2222  2239
=!QUIT   Abs  927962 #E28DA -  2470  2542
=!RCL    Abs  923102 #E15DE -   301   324
=!RDN    Abs  923692 #E182C -   550   575
=!ROLL   Abs  927675 #E27BB -  2378  2423
=!ROT    Abs  924016 #E1970 -   693   715
=!RP!    Abs  923949 #E192D -   656   674
=!RP0    Abs  924255 #E1A5F -   816   831
=!RP@    Abs  923921 #E1911 -   640   655
=!RUP    Abs  923626 #E17EA -   529   549
=!S->D   Abs  927211 #E25EB -  2184  2205
=!S<&    Abs  922373 #E1305 -    36    87
=!S>&    Abs  922558 #E13BE -    88   140
=!SCI    Abs  923293 #E169D -   391   405
=!SIGN   Abs  928767 #E2BFF -  2740  2787
=!SIN    Abs  923351 #E16D7 -   421   434
=!SP!    Abs  923888 #E18F0 -   622   639
=!SP0    Abs  924226 #E1A42 -   801   815
=!SP@    Abs  923773 #E187D -   576   594
=!SPAN   Abs  926384 #E22B0 -  1811  1826
=!SQRTX  Abs  926151 #E21C7 -  1717  1733
=!STD    Abs  923170 #E1622 -   325   340
=!STO    Abs  923047 #E15A7 -   275   300
=!STR$   Abs  929417 #E2E89 -  2985  3069
=!SUB$   Abs  929110 #E2D56 -  2884  2984
=!SWAP   Abs  926927 #E24CF -  2041  2061
=!TAN    Abs  923440 #E1730 -   456   469
=!THEN   Abs  927344 #E2670 -  2261  2291
=!TIB    Abs  924284 #E1A7C -   832   853
=!TYPE   Abs  928904 #E2C88 -  2806  2850
=!U*     Abs  922939 #E153B -   217   258
=!USE    Abs  924328 #E1AA8 -   854   870
=!VAL    Abs  925717 #E2015 -  1515  1536
=!WARN   Abs  927251 #E2613 -  2206  2221
=!WORD   Abs  928176 #E29B0 -  2543  2616
=!X#Y?   Abs  926032 #E2150 -  1658  1716
=!X<>Y   Abs  926614 #E2396 -  1906  1928
=!X<Y?   Abs  925885 #E20BD -  1613  1621
=!X>Y?   Abs  925946 #E20FA -  1632  1640
=!XE0?   Abs  925915 #E20DB -  1622  1631
=!XEY?   Abs  925976 #E2118 -  1641  1657
=!XOR    Abs  924099 #E19C3 -   736   761
=!X^2    Abs  923475 #E1753 -   470   486
=!Y^X    Abs  923554 #E17A2 -   501   514
=!e^X    Abs  923591 #E17C7 -   515   528
=#TIB    Abs  926363 #E229B -  1797
=$#TIB   Abs  926368 #E22A0 -  1799  1797
=$1/X    Abs  923532 #E178C -   491   490
=$10^X   Abs  926694 #E23E6 -  1933  1932
=$2DUP   Abs  927066 #E255A -  2110  2108
 $?DUP   Abs  928746 #E2BEA -  2725  2724
=$ABS    Abs  924195 #E1A23 -   783   781  1740
=$ACOS   Abs  926551 #E2357 -  1882  1881
=$AND    Abs  924076 #E19AC -   720   719
```



```
=$ASIN   Abs  926512 #E2330 -  1868  1867
=$ATAN   Abs  926590 #E237E -  1896  1895
=$BASE   Abs  927328 #E2660 -  2245  2243
=$BLK    Abs  924370 #E1AD2 -   876   874
=$BYE    Abs  924492 #E1B4C -   966   964
 $CHS    Abs  922904 #E1518 -   197   196
=$COS    Abs  923399 #E1707 -   439   438
=$CR?    Abs  928117 #E2975 -  2512  2511
=$CSP    Abs  924447 #E1B1F -   930   929
=$DABS   Abs  927165 #E25BD -  2159  2157
 $DNGAT  Ext                 -  3002
=$DPL    Abs  924415 #E1AFF -   906   905
=$DROP   Abs  926917 #E24C5 -  2029  2027
=$DUP    Abs  924000 #E1960 -   680   678  1739
=$EMIT   Abs  928496 #E2AF0 -  2622  2620
=$ENG    Abs  923335 #E16C7 -   410   409
=$EOF    Abs  925496 #E1F38 -  1426  1424
=$FABS   Abs  925803 #E206B -  1565  1564
=$FILL   Abs  927099 #E257B -  2129  2127
=$FIX    Abs  923210 #E164A -   345   344
=$FLD    Abs  924431 #E1B0F -   918   917
 $FLIT   Abs  926450 #E22F2 -  1848  1846
 $FTOI   Abs  926311 #E2267 -  1775  1773
=$GROW   Abs  926755 #E2423 -  1962  1960
=$HLD    Abs  924463 #E1B2F -   943   941
=$IN     Abs  924399 #E1AEF -   893   891
 $ITOF   Abs  926205 #E21FD -  1739  1737
=$KEY    Abs  924660 #E1BF4 -  1033  1031
=$LIT    Abs  924761 #E1C59 -  1083  1081
 $LOG    Abs  922867 #E14F3 -   181   180
 $M/MOD  Ext                 -  3019
=$NOT    Abs  924167 #E1A07 -   766   765
=$OVER   Abs  927032 #E2538 -  2092  2090
=$PICK   Abs  926982 #E2506 -  2067  2065
=$POS    Abs  925616 #E1FB0 -  1471  1469
=$PREV   Abs  927297 #E2641 -  2227  2225
=$R/W    Abs  925079 #E1D97 -  1239  1232
=$RCL    Abs  923115 #E15EB -   306   304
=$RDN    Abs  923705 #E1839 -   554   553
=$ROT    Abs  924029 #E197D -   698   696
=$RP0    Abs  924268 #E1A6C -   821   819
=$RP@    Abs  923934 #E191E -   644   643
=$RUP    Abs  923639 #E17F7 -   533   532
=$S->D   Abs  927226 #E25FA -  2189  2187
=$SCI    Abs  923306 #E16AA -   395   394
 $SETLV  Abs  927569 #E2751 -  2337  2336
=$SIN    Abs  923364 #E16E4 -   425   424
=$SP!    Abs  923901 #E18FD -   627   626
=$SP0    Abs  924239 #E1A4F -   805   804
=$SP@    Abs  923786 #E188A -   580   579
 $SPAN   Abs  926399 #E22BF -  1816  1814
=$SQRTX  Abs  926166 #E21D6 -  1721  1720
=$STD    Abs  923183 #E162F -   329   328
=$STO    Abs  923060 #E15B4 -   280   278
 $STR$   Abs  929462 #E2EB6 -  2998  2996
```

```
=$SWAP   Abs  926942 #E24DE -  2046  2044
=$TAN    Abs  923453 #E173D -   460   459
=$TIB    Abs  924297 #E1A89 -   837   835
=$TYPE   Abs  928919 #E2C97 -  2811  2809
=$U*     Abs  922952 #E1548 -   222   220
=$USE    Abs  924341 #E1AB5 -   859   857
=$WARN   Abs  927266 #E2622 -  2211  2209
=$X*Y?   Abs  926047 #E215F -  1662  1661
=$X<>Y   Abs  926629 #E23A5 -  1910  1909
 $X<Y?   Abs  925900 #E20CC -  1617  1616
 $X>Y?   Abs  925961 #E2109 -  1636  1635
=$XDO    Abs  924780 #E1C6C -  1097  1096
 $XE0?   Abs  925930 #E20EA -  1626  1625
=$XEY?   Abs  925991 #E2127 -  1645  1644
=$XOR    Abs  924112 #E19D0 -   741   739
=$X^2    Abs  923488 #E1760 -   474   473
=$Y^X    Abs  923567 #E17AF -   505   504
=$e^X    Abs  923604 #E17D4 -   519   518
=/MOD    Abs  928636 #E2B7C -  2685  1126
=1/X     Abs  923527 #E1787 -   490
 1/X15   Ext               -   492
=10^X    Abs  926689 #E23E1 -  1932
 2DROP   Ext               -  2408  2556  2557  2964
=2DUP    Abs  927061 #E2555 -  2108    70   127  1170  1191  2552  2920  2944
                            3077
=4N@     Abs  922776 #E1498 -   145
 <>      Ext               -  2871
 >R      Ext               -    49    59    60    61    98   112   116  1149
                            2568  2686  2913
 ?COMP   Ext               -   263  2266  2297  2323  2793
=?DUP    Abs  928741 #E2BE5 -  2724
 ?PAIR   Ext               -  2268  2299  2325
 ?STK    Ext               -  2486
=@RP!    Abs  923962 #E193A -   661   660
 AB01    Abs  924219 #E1A3B -   792   788
 ABORTW  Ext               -    56   106  2385  2896
 ABRT"X  Ext               -  1147  2765
=ABS     Abs  924190 #E1A1E -   781
=ACOS    Abs  926546 #E2352 -  1881
 ACOS15  Ext               -  1883
 AD1     Abs  926816 #E2460 -  1980  1985
 ADD     Ext               -    44    63    96   118  1069  1213  2395  2572
                            2858  2940
=AND     Abs  924071 #E19A7 -   719   149  2445  2909
 ANOTH   Abs  925131 #E1DCB -  1259  1326
=ASC     Abs  925763 #E2043 -  1541
=ASIN    Abs  926507 #E232B -  1867
 ASIN15  Ext               -  1869
 AT      Ext               -   146  2398  2431  2432  2488  2490  2591  2670
                            2710  2763  2857  2861  2867  2870
=ATAN    Abs  926585 #E2379 -  1895
 ATAN15  Ext               -  1897
 BACK2B  Ext               -  1367  1444  1450
=BASE    Abs  927323 #E265B -  2243   165
=BLK     Abs  924365 #E1ACD -   874  2476  2590  2599
```



```
=BRACT   Abs  923022 #E158E -   262
 BRNCH   Ext               -  2301  2449  2507  2526  2561  2605  2960
 BSIZE   Ext               -  2856
=BYE     Abs  924487 #E1B47 -   964
=BYE1    Abs  924516 #E1B64 -   974   966
 C!      Ext               -    72   130  2571  2711  2950  2952  3078  3080
                            3083
=C@+     Abs  923813 #E18A5 -   598
 C@0     Abs  923873 #E18E1 -   610   601
=CASE    Abs  928869 #E2C65 -  2792
 CAT     Ext               -    53   104   608  1543  2586
=CHK     Abs  926858 #E248A -  2001  1962
=CHR$    Abs  929668 #E2F84 -  3074
=CHS     Abs  922899 #E1513 -   196
 CKINF-  Ext               -  2636
 CMOV>   Ext               -   120
 CMOVE   Ext               -    67   124  2956
=CMPST   Abs  926060 #E216C -  1686  1627
 CMPST+  Abs  926102 #E2196 -  1697  1694
=CMPST-  Abs  926057 #E2169 -  1685  1617  1636  1645  1662
 CMPSUB  Abs  926120 #E21A8 -  1704  1692  1699
 COMMA   Ext               -  2304  2330
 COMPL   Ext               -  2300  2326
 CONTX   Ext               -  2430
=COS     Abs  923394 #E1702 -   438
 COS15   Ext               -   440
=CR?     Abs  928112 #E2970 -  2511  2479
 CRDONE  Abs  928148 #E2994 -  2519  2517
=CRLF    Abs  925857 #E20A1 -  1593
=CSP     Abs  924442 #E1B1A -   929  2762
 Clear   Ext               -  2516
=D.R     Abs  924866 #E1CC2 -  1143
 D0+2RD  Ext               -  1278  1300  1311  1317
=DABS    Abs  927160 #E25B8 -  2157
 DEFADR  Ext               -  1039
 DEPTH   Ext               -  1144  2502  2889
 DGT01   Abs  923268 #E1684 -   373   371
 DGT1    Abs  923258 #E167A -   369   367
 DOCOL   Ext               -    39    91   145   162   262   598  1065  1125
                            1143  1169  1190  1207  1519  1541  1593  1830
                            2265  2296  2322  2381  2428  2474  2525  2547
                            2668  2685  2704  2744  2760  2792  2836  2855
                            2888  2988  3074  3105
 DOT     Ext               -  2503
 DOUSE   Ext               -   805   821   859   876   893   906   918   930
                             943  1799  1816  2211  2227  2245
 DP      Ext               -  2669
=DPL     Abs  924410 #E1AFA -   905
=DROP    Abs  926912 #E24C0 -  2027  1127  1176  1197  1520  1542  1832  2454
                            2604  2608  2865  2963
 DSPDGT  Ext               -   373
 DSPSTA  Ext               -  2513
 DSPSUB  Abs  928532 #E2B14 -  2636  2630  2823
=DUP     Abs  923995 #E195B -   678    40    48    92    97   111   113   115
                             599  2382  2388  2396  2429  2548  2567  2584
```

```
                                  2796  2859  2868  2929
=ELSE    Abs  927419 #E26BB -     2296
=EMIT    Abs  928491 #E2AEB -     2620
 ENCL    Ext              -       2551
=END$    Abs  929753 #E2FD9 -     3105
 ENDR    Abs  925196 #E1E0C -     1284  1264  1276  1290
=ENG     Abs  923330 #E16C2 -      409
=EOF     Abs  925491 #E1F33 -     1424  2594
 EQUAL   Ext              -       2862
 EXP15   Ext              -        520
=EXXIT   Abs  926425 #E22D9 -     1830
=FABS    Abs  925798 #E2066 -     1564
 FAIL    Abs  925693 #E1FFD -     1498  1477  1483
 FAL1    Abs  925583 #E1F8F -     1453  1448
 FALSE   Abs  925574 #E1F86 -     1450  1443
 FEND    Ext              -        183   427   441   462   479   493   507   521
                                  1723  1870  1884  1898  1943
 FIBAD-  Ext              -       1252  1429
 FIL0    Abs  927126 #E2596 -     2139  2142
 FIL1    Abs  927140 #E25A4 -     2144  2135
=FILL    Abs  927094 #E2576 -     2127
=FIND    Abs  927842 #E2862 -     2428
 FIND>   Ext              -       2433
 FINLIN  Ext              -       2518
 FIRST   Ext              -       2866
 FIVE-   Ext              -       2397  2401
=FIX     Abs  923205 #E1645 -      344
=FLD     Abs  924426 #E1B0A -      917
=FLIT    Abs  926445 #E22ED -     1846
 FLTOH   Ext              -       1778
 FND0    Abs  927947 #E28CB -     2457  2437
 FND01   Abs  927922 #E28B2 -     2451  2447
 FND02   Abs  927932 #E28BC -     2453  2450
 FND03   Abs  927952 #E28D0 -     2458
 FOUR+   Ext              -       2954  2958  3082  3085
 FSTRT   Ext              -        505
=FTOI    Abs  926306 #E2262 -     1773
 GETFP   Ext              -        315   541   566  1037  1285  1453  1502  1652
                                  1749  1758  1780  1856  1987  2012  2355  2519
                                  2632  2825
 GETPTR  Ext              -       1348  1430
 GETX+L  Ext              -        448
=GROW    Abs  926750 #E241E -     1960
 GT      Ext              -        105  1171  2553  2891
 GT1     Abs  924579 #E1BA3 -      993   991
 GT2     Abs  924597 #E1BB5 -     1007  1005
 GT3     Abs  924617 #E1BC9 -     1012  1009
 GT4     Abs  924640 #E1BE0 -     1018  1014
=GTS13   Abs  924554 #E1B8A -      986  1002
=GTSTAT  Abs  924581 #E1BA5 -     1002   975
 HDFLT   Ext              -       1747
=HERE    Abs  928601 #E2B59 -     2668  1066  2269  2302  2328  2331  2559  2570
                                  2573  2578
=HEX     Abs  922819 #E14C3 -      162
=HLD     Abs  924458 #E1B2A -      941  2707  2709
```



```
=HOLD    Abs  928681 #E2BA9 -     2704  2750
=IN      Abs  924394 #E1AEA -      891  2563  2602  2838
 INTRP   Ext              -       2485
=ITOF    Abs  926200 #E21F8 -     1737
 ITOF0   Abs  926276 #E2244 -     1757  1753
=KEY     Abs  924655 #E1BEF -     1031
 KEYRD   Ext              -       1034
 LBRAC   Ext              -       2478
 LGT15   Ext              -        182
 LIMIT   Ext              -       2860
=LIT     Abs  924756 #E1C54 -     1081    41    45    50    93    99   147   163
                                  1067  1210  2391  2443  2451  2705  2748  2794
                                  2907  2946
=LOG     Abs  922862 #E14EE -      180
=LOOP    Abs  927504 #E2710 -     2322
 LT      Ext              -        55  1146  1192  2384  2922
 LTZ     Ext              -       2745
 M/      Ext              -       2689
 MAINLP  Ext              -        970
 MAINST  Ext              -       1967
 MATCH   Abs  925695 #E1FFF -     1501  1486
=MAX     Abs  924999 #E1D47 -     1190  2916  2948
 MAX1    Abs  925029 #E1D65 -     1197  1194
 MEMCKL  Ext              -       1964
=MIN     Abs  924946 #E1D12 -     1169  2911
 MIN1    Abs  924976 #E1D30 -     1176  1173
 MINUS   Ext              -        52  1153  2271  2329  2566  2764  2932
=MOD     Abs  924833 #E1CA1 -     1125
 MP2-15  Ext              -        478
 MTHSTK  Ext              -       1688
 MULT    Ext              -       2393
 MVMEM+  Ext              -       1972
 NMOVE   Ext              -       2577
 NONE    Abs  927627 #E278B -     2355  2348
 NORED   Abs  925181 #E1DFD -     1278  1281
 NORED0  Abs  925188 #E1E04 -     1280  1277
=NOT     Abs  924162 #E1A02 -      765
 NOTEND  Abs  925161 #E1DE9 -     1271  1266
 NULL$   Ext              -       2965
 NUMB    Ext              -       1525
=NUMST   Abs  923416 #E1718 -      446   181   425   439   460   474   491   519
                                  1721  1868  1882  1896  1933
 NoCont  Ext              -       1003  1006  1017
 OKFLG   Ext              -       2487
 ONE     Ext              -       1208  2383  2448  2598  2895  2915  2990  3075
                                  3086
 ONE+    Ext              -       2387  2933
 ONE-    Ext              -        606  2937
 OR      Ext              -       2491
=OVER    Abs  927027 #E2533 -     2090    54   602  1152  2270  2399  2403  2565
                                  2942  2945  2991  3106
=PAD     Abs  924726 #E1C36 -     1065  2943  2949  2957  3076  3081  3084
=PBUF    Abs  929015 #E2CF7 -     2855
 PBUF0   Abs  929080 #E2D38 -     2868  2864
 PDOTQ   Ext              -       2499  2504
```

```
=PFA     Abs  925039 #E1D6F -  1207
=PICK    Abs  926977 #E2501 -  2065   43   47   95  101 2389
=POS     Abs  925611 #E1FAB -  1469
=PREV    Abs  927292 #E263C -  2225 2869
 PSTOR   Ext                -  2564 2600 2708 2839
=PT1+    Abs  928975 #E2CCF -  2836
=QCSP    Abs  928817 #E2C31 -  2760
 QUERY   Ext                -  2484
=QUIT    Abs  927972 #E28E4 -  2474
 QUIT0   Abs  928102 #E2966 -  2507 2498
 QUIT00  Abs  927997 #E28FD -  2479 2508
 QUIT01  Abs  928002 #E2902 -  2483
 QUITX   Abs  928062 #E293E -  2499 2527
 QUOTC   Ext                -  1594
=R/W     Abs  925074 #E1D92 -  1232
 R>      Ext                -    64   66   68   69  121  125  126 1151
                             1831 2575 2688 2918
 R@      Ext                -   122
=RCL     Abs  923110 #E15E6 -   304
 RDBYT   Abs  925234 #E1E32 -  1297 1305
 RDBYT0  Abs  925250 #E1E42 -  1302 1295
 RDLNAS  Ext                -  1360 1439
=RDN     Abs  923700 #E1834 -   553
 READIT  Abs  925207 #E1E17 -  1289 1274
 READR   Abs  925094 #E1DA6 -  1244 1242
 REDREC  Abs  925345 #E1EA1 -  1346 1275 1289
=REND2   Abs  929773 #E2FED -  3110
 REPOS   Abs  925313 #E1E81 -  1328 1267
 RESET   Ext                -  1484
 REWIND  Ext                -  1331
=ROLL    Abs  927685 #E27C5 -  2381
 ROLL1   Abs  927757 #E280D -  2396 2407
=ROT     Abs  924024 #E1978 -   696  109  110  114  119  603  605  607
                             2453 2900 2930 2935 2936
=RP!     Abs  923957 #E1935 -   660 2483
=RP0     Abs  924263 #E1A67 -   819
=RP@     Abs  923929 #E1919 -   643
 RREC10  Abs  925467 #E1F1B -  1402 1394
 RREC5   Abs  925441 #E1F01 -  1386 1363
=RUP     Abs  923634 #E17F2 -   532
=S->D    Abs  927221 #E25F5 -  2187 2687
=S<&     Abs  922381 #E130D -    39
=S>&     Abs  922566 #E13C6 -    91
 SAME    Ext                -  1485
 SAVEFP  Ext                -   306  447  533  554 1033 1244 1426 1471
                             1686 1743 1775 1851 2001 2339 2512 2622
                             2821
=SCI     Abs  923301 #E16A5 -   394
 SD0     Abs  927238 #E2606 -  2194 2192
 SEMI    Ext                -    74  131  150  167  265  609  611 1070
                             1128 1156 1177 1198 1214 1526 1544 1599
                             1833 2275 2309 2333 2409 2455 2458 2609
                             2671 2690 2712 2751 2770 2797 2840 2872
                             2966 2994 3087 3108
 SENDWD  Ext                -  2641
```



```
 SETDGT  Abs  923232 #E1660 -   360  345  395  410
 SETFMT  Ext                -   349
 SETGET  Abs  923217 #E1651 -   349  331  397  412
 SETL0   Abs  927602 #E2772 -  2345 2368
 SETL1   Abs  927633 #E2791 -  2358 2354
=SETLEV  Abs  927564 #E274C -  2336 2332
 SETSTR  Ext                -  1472
 SFLAGS  Ext                -  1011 1016
=SIGN    Abs  928777 #E2C09 -  2744
 SIGN00  Abs  928812 #E2C2C -  2751 2747
 SIGTST  Ext                -  1705
=SIN     Abs  923359 #E16DF -   424
 SIN15   Ext                -   426
=SP!     Abs  923896 #E18F8 -   626
=SP0     Abs  924234 #E1A4A -   804
=SP@     Abs  923781 #E1885 -   579 2394 2402 2761
 SPACS   Ext                -  1154
=SPAN    Abs  926394 #E22BA -  1814
 SQR15   Ext                -  1722
=SQRTX   Abs  926161 #E21D1 -  1720
 STATE   Ext                -  2489
=STD     Abs  923178 #E162A -   328
 STKDRP  Ext                -   558
 STKLFT  Ext                -   311  537 1745 1852
 STMTR0  Ext                -  2646
=STO     Abs  923055 #E15AF -   278
 STORE   Ext                -   166 2273 2400 2477 2560 2603
=STR$    Abs  929457 #E2EB1 -  2996 1150 2989
=STR$U   Abs  929427 #E2E93 -  2988
 STR00   Abs  929479 #E2EC7 -  3003 3001
 STR01   Abs  929525 #E2EF5 -  3016 3042
 STR02   Abs  929570 #E2F22 -  3030 3028
 STR03   Abs  929587 #E2F33 -  3037 3034
 STR04   Abs  929622 #E2F56 -  3049 3045
 STRM    Ext                -  2549
=SUB$    Abs  929120 #E2D60 -  2888 2993 3107
 SUB$1   Abs  929167 #E2D8F -  2900 2894
 SUB01   Abs  929392 #E2E70 -  2963 2927
 SUB02   Abs  929407 #E2E7F -  2966 2961
=SWAP    Abs  926937 #E24D9 -  2044   65   73  123  128 1174 1195 2272
                             2305 2390 2550 2607 2921 2928 2931 2955
                             2959
=TAN     Abs  923448 #E1738 -   459
 TAN15   Ext                -   461
=THEN    Abs  927354 #E267A -  2265 2307
 THREE   Ext                -  1145 2324 2890
=TIB     Abs  924292 #E1A84 -   835
 TIC     Ext                -   264
 TP0     Abs  928970 #E2CCA -  2827 2813 2815
 TR0     Abs  926013 #E213D -  1652 1650
 TRA0    Abs  925273 #E1E59 -  1314 1282 1303
 TRA00   Abs  925287 #E1E67 -  1318 1316
 TRA1    Abs  925300 #E1E74 -  1323 1335
 TRASH   Abs  925261 #E1E4D -  1311 1313
 TRAVS   Ext                -  1209
```

```
 TRU      Abs  925567 #E1F7F -  1446  1435
 TRY      Abs  925641 #E1FC9 -  1479  1495
 TWO      Ext         -      2267  2298  2306  2308  2837
 TWO*     Ext         -        62   117  2938
 TWO+     Ext         -       604  2574  2576  2585  2951  3079
 TWO-     Ext         -        71   102   103   129  1521
 TWO/     Ext         -      2569
=TYPE     Abs  928914 #E2C92 -  2809  1155
=U*       Abs  922947 #E1543 -   220
 U*0      Abs  922986 #E156A -   235   233
 U*1      Abs  922972 #E155C -   230   237
 U<       Ext         -      2404
 UPCPOS   Ext         -      1320
=USE      Abs  924336 #E1AB0 -   857
=VAL      Abs  925725 #E201D -  1519
=WAKE     Abs  928156 #E299C -  2525
=WARN     Abs  927261 #E261D -  2209
 WD       Abs  928266 #E2A0A -  2563  2555
 WD0      Abs  928341 #E2A55 -  2578  2562
 WD00     Abs  928461 #E2ACD -  2607  2589  2593  2597
=WORD     Abs  928186 #E29BA -  2547
 WR0      Abs  926882 #E24A2 -  2010  1966  2006
=WR1      Abs  926884 #E24A4 -  2012
 WRD0     Abs  928191 #E29BF -  2548  2606
 WRITE    Abs  925202 #E1E12 -  1286  1243
=X#Y?     Abs  926042 #E215A -  1661
 X1YY     Abs  925995 #E212B -  1646  1629
=X<>Y     Abs  926624 #E23A0 -  1909
=X<Y?     Abs  925895 #E20C7 -  1616
=X>Y?     Abs  925956 #E2104 -  1635
=XD0      Abs  924775 #E1C67 -  1096
=XE0?     Abs  925925 #E20E5 -  1625
=XEY?     Abs  925986 #E2122 -  1644
 XLOOP    Ext         -      2327
=XOR      Abs  924107 #E19CB -   739
=XXYY     Abs  925997 #E212D -  1647  1619  1638  1664
=X^2      Abs  923483 #E175B -   473
 YX2-15   Ext         -       506  1942
=Y^X      Abs  923562 #E17AA -   504
 ZBRNH    Ext         -       600  1172  1193  2406  2436  2446  2497  2554
                              2588  2592  2596  2746  2863  2893  2926
 ZEQ      Ext         -      2405  2496  2587  2595  2892  2924
 ZERO     Ext         -       610  2303  2457  2475  2558  2601
=adj      Abs  926798 #E244E -  1974
=e^X      Abs  923599 #E17CF -   518
=fFAIL    Abs  925693 #E1FFD -  1497
=fMATCH   Abs  925695 #E1FFF -  1500
 flPRGM   Ext         -      1010
 flSUSP   Ext         -      1015
 uMODES   Ext         -       446
 uRES12   Ext         -      1707
 uTEST    Ext         -      1647
```



```
Input Parameters

  Source file name is MR&FT2

  Listing file name is MR/FT2

  Object file name is MR%FT2::65

                                        111111
                                0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News
```



```
 1                TITLE  FT3:_5_CHAR_WORDS
 2                RDSYMB MR%GTO
 3
 4          *     ABS    REND2
 5
 6 E3050          ABS    #E3050
 7
 8          pENTER EQU    #12      !!!!!TEMPORARY MEASURE!!!
 9          MFERsp EQU    #940D
10
11
12          ****************************************************
13          *
14          * MR&FT3     <840608.1403>
15          *
16          *
17          *  START OF 5 CHAR WORDS
18          * ADD ADDITIONAL WORDS AT THIS END
19          *
20          ****************************************************
```

```
                        EJECT
                *************************************************************
                *
                * NFILL (addr u nib --- ) fill memory beginning at addr with
                *       n copies of nib, if quantity is  = 0 then
                *       take no action
                *
                *************************************************************

E3050 0000      NO MORE 5 CHAR WORDS    CON(5)  0
      0
E3055 58        =!NFILL CON(2)  #85             NFILL
E3057 E464      NFIL    NIBASC  \NFIL\
      94C4
E305F CC                CON(2)  \L\+#80
E3061 6603      =NFILL  CON(5)  =$NFILL
      F

E3066 143       =$NFILL A=DAT1  A               READ nib  TO A
E3069 174               D1=D1+  5
E306C 147               C=DAT1  A               READ U
E306F 174               D1=D1+  5
E3072 D7                D=C     A               COPY U TO D
E3074 147               C=DAT1  A               READ ADDR
E3077 174               D1=D1+  5               RETURN NOTHING ON STACK
E307A 8AB               ?D=0    A               ANY TO DO?
E307D 61                GOYES   NFIL1           NO, GO HOME
E307F CF                D=D-1   A               PRIME FOR CARRY TEST
E3081 136               CD0EX                   D0=ADDR,C=I
E3084 1580      NFIL0   DAT0=A  1               STORE nib  AT ADDR
E3088 160               D0=D0+  1               INCREMENT ADDR
E308B CF                D=D-1   A               DECREMENT COUNT
E308D 56F               GONC    NFIL0           DO ANOTHER IF NO CARRY
E3090 134               D0=C                    RESTORE I TO D0
E3093 03        NFIL1   RTNCC
```



```
                        EJECT
                *************************************************
                *
                * FSTR$  ( --- str)  This word creates a string
                *       at PAD to represent the current contents
                *       of the X register.  The string is created
                *       according to the current display format.
                *
                *************************************************

E3095 5503      !NFILL
      E
E309A 58        =!FSTR$ CON(2)  #85
E309C 6435      FSTR    NIBASC  \FSTR\
      4525
E30A4 4A                CON(2)  \$\+#80
E30A6 0000      =FSTR$  CON(5)  =DOCOL
      0
E30AB 0000              CON(5)  =XSTR$          ( addr cnt )
      0
E30B0 0000              CON(5)  =ONE            ( addr cnt 1 )
      0
E30B5 0000              CON(5)  =OVER           ( addr cnt 1 cnt )
      0
E30BA 0000              CON(5)  =SUB$           ( addr cnt of str at pad )
      0
E30BF 0000              CON(5)  =SEMI
      0
```

```
                        EJECT
                *****************************************
                *
                * VARID         ( -- N )
                *
                * System user variable used by assembler.
                * The general purpose buffer identified
                * by the contents of this variable will
                * be preserved when the system cleans up
                * memory.
                *
                *****************************************
E30C4 A903E             CON(5)  !FSTR$
E30C9 58        =!VARID CON(2)  #85
E30CB 6514              NIBASC  \VARI\
      2594
E30D3 4C                CON(2)  \D\+#80
E30D5 AD03E     =VARID  CON(5)  =$VARID
E30DA 8E00      =$VARID GOSUBL  =DOUSE
      00
E30E0 34CF2             CON(5)  (=oVARID)
```



```
                        EJECT
                *****************************************
                *
                * FDROP   ( --- )  This word does a stack
                *       drop on the floating point stack.
                *       Its purpose is to replace CLX by
                *       allowing you to drop the contents
                *       of the X register.
                *
                *****************************************
E30E5 9C03E             CON(5)  =!VARID
E30EA 58        =!FDROP CON(2)  #85
E30EC 6444              NIBASC  \FDRO\
      25F4
E30F4 0D                CON(2)  \P\+#80
E30F6 BF03E     =FROP   CON(5)  $FDROP
E30FB 8E00      $FDROP  GOSUBL  =SAVEFP         SAVE FORTH ENVIRONMENT
      00
E3101 1B0CBF2           D0=(5)  =oLASTX
E3108 1567              C=DAT0  W
E310C 10B               R3=C                    save LASTX
E310F 8E00              GOSUBL  =STKDRP         DROP STACK
      00
E3115 1B0CBF2           D0=(5)  =oLASTX
E311C 11B               C=R3
E311F 1547              DAT0=C  W               restore old LASTX
E3123 8C00              GOLONG  =GETFP          RESTORE FORTH ENVIRONMENT
      00
```

```
                    EJECT
                *****************************************************
                *
                * Renamed FINDF (from FFIND)
                * 12/18/83
                *
                * FINDF         ( str -- addr,or 0 if not found )
                *
                * Return the address of the file in RAM with
                * given name.
                *
                *****************************************************
E3129 AE03      E       CON(5)  !FDROP
E312E 58        =!FFIND CON(2)  #85
E3130 6494      E444    NIBASC  \FIND\
E3138 6C                CON(2)  \F\+#80
E313A 0000      0       =FFIND  CON(5)  =DOCOL
E313F 0000      0               CON(5)  =SETNAM         PUT NAME IN STMTR0 FOR FSYNTX
E3144 0000      0               CON(5)  =FSYNTX         CHECK SYNTAX & SET UP FNAME

                * essentially calls routine FSPECx -- if name is legal it
                * is returned blank-filled; this is stored in FNAME

E3149 0000      0               CON(5)  =ZBRNH          IF OK
E314E 4100      0               CON(5)  (FSX)-*
E3153 0000      0               CON(5)  =FNAME          PROPER FNAME ON STACK
E3158 AA13      E               CON(5)  =FILFND      return addr of file
E315D 0000      0               CON(5)  =SEMI

E3162 0000      0       FSX     CON(5)  =FNAME          addr of 8 char name
E3167 0000      0               CON(5)  =LIT
E316C 8000      0               CON(5)  8
E3171 D713      E               CON(5)  =ABORTN         move name to HERE for msg
E3176 22E3      E               CON(5)  =ABORTW
E317B 31                        CON(2)  =eBFIL

                * This is a tricky way to make the MSG: "FTH ERR: <str>
                * REST OF MESSAGE" ; it assumes a string and moves it to
                * HERE as a counted string for ABORTW
```



```
E317D 0000      0       =ABORTN CON(5)  =DOCOL
E3182 0000      0               CON(5)  =DUP            LEN
E3187 0000      0               CON(5)  =HERE
E318C 0000      0               CON(5)  =C!             PUT IN LEN
E3191 0000      0               CON(5)  =HERE
E3196 0000      0               CON(5)  =TWO+
E319B 1454      E               CON(5)  =SMOVE          MOVE STR TO HERE FOR ABORTW
E31A0 0000      0               CON(5)  =ONE            TRUE, GO ABORT
E31A5 0000      0               CON(5)  =SEMI


E31AA FA13      E       =FILFND CON(5)  =$FILF
E31AF 8E00      00      =$FILF  GOSUBL  =SAVEFP
E31B5 143               A=DAT1  A               GET ADDR OF FNAME
E31B8 131               D1=A
E31BB 1537              A=DAT1  W               GET NAME
E31BF 8F00      000             GOSBVL  =FINDF          FIND IT!
E31C6 841               ST=0    1               USE S1 FOR FOUND FLAG
E31C9 450               GOC     FF1             CARRY SET=NOT FOUND
E31CC 851               ST=1    1               FLAG FOUND
E31CF 133       FF1     AD1EX                   ADDR TO A
E31D2 8E00      00              GOSUBL  =GETFP
E31D8 871               ?ST=1   1
E31DB 40                GOYES   FF2             IF FOUND LEAVE ADDR IN A
E31DD D0                A=0     A
E31DF 141       FF2     DAT1=A  A               REPLACE FNAME BY ADDR
E31E2 03                RTNCC
```

```
                        EJECT
                *******************************************************
                *
                *  X=>Y?  X<>Y?  X<=Y?    ( --- t/f)
                *
                * These words compare the contents of the X & Y registers
                * in the floating point stack and if the specified test
                * is true they return a true flag on the integer data stack
                * else they return a false flag.
                *
                *******************************************************
E31E4 E213E     CON(5)  =!FFIND
E31E9 58    =!X<=Y? CON(2)  #85
E31EB 85C3D395  NIBASC  \X<=Y\
E31F3 FB        CON(2)  \?\+#80
E31F5 AF13E =X<=Y?  CON(5)  $X<=Y?
E31FA 8E0000 $X<=Y?  GOSUBL  =CMPST-
E3200 23        P=      3           P INDICATES TEST
E3202 8C0000    GOLONG  =XXYY       FINISH IN MR&FT2

E3208 9E13E     CON(5)  !X<=Y?
E320D 58    =!X=>Y? CON(2)  #85
E320F 85E3D395  NIBASC  \X>=Y\
E3217 FB        CON(2)  \?\+#80
E3219 E123E =X=>Y?  CON(5)  $X=>Y?
E321E 8E0000 $X=>Y?  GOSUBL  =CMPST-
E3224 26        P=      6           P INDICATES TEST
E3226 8C0000    GOLONG  =XXYY       FINISH IN MR&FT2
```



```
                        EJECT
                *******************************************************
                *
                * CONBF  (n1 n2 --- t/f) This word will contract
                *       the I/O buffer identified by the ID#, n2,
                *       by the number of nibs, n1, specified.  If
                *       successful it returns a true flag, else
                *       it returns a false flag.
                *
                * n1 changed to mean nibs instead of bytes
                * 12/12/83 B.M.
                *******************************************************
E322C D023E     CON(5)  !X=>Y?
E3231 58    =!IOCON CON(2)  #85
E3233 34F4E424  NIBASC  \CONB\
E323B 6C        CON(2)  \F\+#80
E323D 2423E =IOCON  CON(5)  $IOCON

E3242 D0    $IOCON  A=0     A
E3244 E4        A=A+1   A               CONTRACT FLAG
E3246 5A1       GONC    IOECO

                *******************************************
                *
                * EXPBF  ( n1 n2 --- t/f)  This word will expand
                *       the I/O buffer identified by the ID#, n2,
                *       by the number of nibs, n1, specified.  If
                *       successful it returns a true flag, else
                *       it returns a false flag.
                *
                * n1 changed to mean nibs instead of bytes
                * 12/12/83
                *******************************************
E3249 1323E     CON(5)  !IOCON
E324E 58    =!IOEXP CON(2)  #85
E3250 54850524  NIBASC  \EXPB\
E3258 6C        CON(2)  \F\+#80
E325A F523E =IOEXP  CON(5)  $IOEXP

E325F D0    $IOEXP  A=0     A               EXPAND FLAG

E3261 174   IOECO   D1=D1+  5
E3264 8E0000        GOSUBL  =SAVEFP         SAVE FORTH ENVIRONMENT
                *
                * 2/6/84 changed from ABEX  A; A=DAT1 A to B=A  A;
                * A=DAT1 X, ensures that when we do a B=A  A,  A(A)
                * = 0 or 1 which is overwritten when we do a A=DAT1 X
                *
```

```
261 E326A D8          B=A     A          save exp/con flag in B(A)
262 E326C 1533        A=DAT1  X          read nib count (FFF max, 2/6/84)
263 E3270 1C4         D1=D1-  5
264 E3273 15F2        C=DAT1  3          READ BUFFER ID#
265 E3277 DC          ABEX    A          NIB COUNT TO B(A)
266 E3279 8AC         ?A#0    A          DOING CONTRACT?
267 E327C B1          GOYES   IOEC1      YES
268 E327E 8F00        GOSBVL  =I/OEXP    EXPAND BUFFER
          000
269 E3285 5B1         GONC    IOFAIL     DIDN'T WORK!
270 E3288 D0   SUCC   A=0     A
271 E328A CC          A=A-1   A          TRUE FLAG
272 E328C 8E00 IOEC2  GOSUBL  =GETFP     RESTORE FORTH ENVIRONMENT
          00
273 E3292 141         DAT1=A  A          WRITE FLAG TO STACK
274 E3295 03          RTNCC
275 E3297 8F00 IOEC1  GOSBVL  =I/OCON    CONTRACT BUFFER
          000
276 E329E 49E         GOC     SUCCE      SUCCESSFUL
277
278 E32A1 D0   IOFAIL A=0     A
279 E32A3 58E         GONC    IOEC2        B.E.T.  (2/6/84)
```



```
280                EJECT
281           ******************************************************
282           *
283           * FLUSH  ( --- ) Unassigns all block buffers.
284           *
285           ******************************************************
286 E32A6 E423          CON(5)  !IOEXP
          E
287 E32AB 58    =!FLUSH CON(2)  #85
288 E32AD 64C4          NIBASC  \FLUS\
          5535
289 E32B5 8C            CON(2)  \H\+#80
290 E32B7 0000  =FLUSH  CON(5)  =DOCOL      :
          0
291 E32BC D383          CON(5)  =FIRST      FIRST      get addr of first
          E
292 E32C1 0000          CON(5)  =AT         @          buffer,
          0
293 E32C6 0000          CON(5)  =DUP                   make a copy
          0
294 E32CB 0000          CON(5)  =PREV       PREV       set PREV=FIRST for
          0
295 E32D0 0000          CON(5)  =STORE      !          +BUF in BEGIN UNTIL
          0
296 E32D5 0000 FLU0     CON(5)  =ZERO       BEGIN 0    set fib field to 0
          0
297 E32DA 0000          CON(5)  =OVER       OVER       to unassign buffer
          0
298 E32DF 0000          CON(5)  =C!         C!
          0
299 E32E4 0000          CON(5)  =PBUF       +BUF
          0
300 E32E9 0000          CON(5)  =ZEQ        0=
          0
301 E32EE 0000          CON(5)  =ZBRNH      UNTIL
          0
302 E32F3 2EFF          CON(5)  (FLU0)-*
          F
303 E32F8 0000          CON(5)  =DROP       DROP
          0
304 E32FD 0000          CON(5)  =SEMI       ;          throw away unwanted buf
          0
```

```
                        EJECT
                *************************************************
                *
                * >BODY  (addr1 --- addr2)  FORTH 83 Standard
                *       This rtne expects that addr1 is the cfa
                *       of a word.  It adds five to it to get the
                *       pfa which it returns as addr2.
                *
                *************************************************
E3302 BA23E     CON(5)  !FLUSH
E3307 58F444    =!>BODY CON(2)  #85
E3309 E324      NIBASC  \>BOD\
E3311 9D        CON(2)  \Y\+#80
E3313 8133E     =>BODY  CON(5)  $>BODY

E3318 20        $>BODY  P=      0
E331A 3450000   LC(5)   5               LOAD C WITH 5
E3321 143       A=DAT1  A               READ CFA
E3324 CA        A=A+C   A               ADD 5 TO GET PFA
E3326 141       DAT1=A  A               WRITE BACK TO STACK
E3329 03        RTNCC                   RETURN TO USER
```



```
                        EJECT
                *************************************************
                *
                * LASTX ( --- ) Puts the contents of LASTX into
                *       X after doing a stack lift.
                *
                *************************************************
E332B 7033E     CON(5) !>BODY
E3330 58        =!LASTX CON(2) #85
E3332 C4143545  NIBASC \LAST\
E333A 8D        CON(2) \X\+#80
E333C 00000     =LASTX  CON(5) =DOCOL           put address of
E3341 00000     CON(5) =LIT             LASTX variable on stack
E3346 0CBF2     CON(5) =oLASTX          then call RCL to do a
E334B 00000     CON(5) =RCL             stack lift and to put
E3350 00000     CON(5) =SEMI            LASTX into X
```

```
342                     EJECT
343                   ***********************************************
344                   *
345                   * ENTER  (addr n c 0 ---  addr n )
346                   *        (addr n --- addr n)
347                   *
348                   *  ENTER  This word will work if the HPIL
349                   *       module is plugged in.  Otherwise it
350                   *       will give an "HPIL ERROR"
351                   *
352                   *  CASE1: 1st parameter on stack is a 0,
353                   *   ENTER will then terminate on receipt of a
354                   *   specific character which it expects to be
355                   *   the second argument on the stack.  The 3rd
356                   *   argument is the maximum number of chars
357                   *   that the user wants to be transferred to the
358                   *   FORTH environment (i.e., a user expects no
359                   *   more than 10 chars from a device before seeing
360                   *   the stop char, however the device sends 1000 chars,
361                   *   if the user had specified 10 as the  max # of chars
362                   *   to be brought into the FORTH environment he is
363                   *   protected from having his memory corrupted).
364                   *   The last parameter is the address at which the
365                   *   string is to be stored in the FORTH environment.
366                   *
367                   *  CASE2: 1st parameter on the stack is non-zero.
368                   *   ENTER assumes that this is the number of chars
369                   *   it should read from the specified device. The 2nd
370                   *   parameter is the address at which the string is
371                   *   to be stored in the FORTH environment.
372                   *
373                   *   The device is specified by the 3 user variables:
374                   *   HPIL-LOOP, SECONDARY and PRIMARY.
375                   *
376                   *   If the user sets system flag -23 (from BASIC)
377                   *   ENTER will terminate with receipt of an ETO.
378                   *
379                   ***********************************************
380 E3355 0333             CON(5) !LASTX
        E
381 E335A 58        =!ENTER CON(2) #85
382 E335C 54E4              NIBASC \ENTE\
        4554
383 E3364 2D                CON(2) \R\+#80
384
385 E3366 0000       =ENTER  CON(5) =DOCOL          :
        0
386 E336B E833              CON(5) =ENT            DO ENTERING
        E
387 E3370 0000              CON(5) =PILERR         REPORT ANY ERROR
        0
388 E3375 0000              CON(5) =SEMI
        0
389
390 E337A 20         ERR    P=     0
```



```
391 E337C 864               ?ST=0  4               plain old error?
392 E337F 80                GOYES  ERR1            yes
393 E3381 3300              LC(4)  =eMEM           otherwise we've run out of memory
        00
394
395 E3387 8D00 ERR1         GOVLNG =BSERR          report error
        000
396
397
398 E338E 3933 =ENT         CON(5) =$ENT
        E
399
400 E3393 8E00 =$ENT        GOSUBL =NIB7           COMPUTE 7 NIB ADDR
        00
401                   *                            FROM HPIL-LOOP, PRIMARY
402                   *                            AND SECONDARY AND RETURN
403                   *                            IT IN A(7), SAVE FORTH
404                   *                            POINTERS & COPY DATA STK PTR
405                   *                            ONTO CPU RTN STK
406
407                   * SET ENTRY CONDITIONS FOR ENTER POLL
408
409 E3399 04                  SETHEX
410 E339B 20                P=     0
411 E339D 847               ST=0   7
412 E33A0 846               ST=0   6
413 E33A3 101               R1=A                   COPY 7 NIB ADDR TO R1
414 E33A6 855               ST=1   5               READ BY BYTE COUNT
415
416 E33A9 07                C=RSTK
417 E33AB 134               D0=C                   RECOVER DATA STK PTR
418 E33AE 142               A=DAT0 A               READ 1ST ITEM
419 E33B1 164               D0=D0+ 5
420 E33B4 8AC               ?A#0   A               GET BYTE COUNT?
421 E33B7 61                GOYES  ENT0            NEXT PART
422
423                   *                            we have to ensure that we
424                   *                            will read an arbitrary #
425                   *                            of chars; we do this by
426                   *                            setting the nibble at
427                   *                            S-R0-3 to 0; A=0 from read
428
429 E33B9 1F00               D1=(5)  =S-R0-3
        000
430 E33C0 1590              DAT1=A  1
431
432 E33C4 14A               A=DAT0 B               read stop char from data stk
433 E33C7 169               D0=D0+ 10              point to addr not max count
434 E33CA 845               ST=0   5               not reading for byte count
435
436 E33CD 136  ENT0          CD0EX                  SAVE UPDATED FORTH DATA
437 E33D0 06                RSTK=C                 STK PTR ON CPU RTN STK
438 E33D2 1B00              D0=(5) =FORSTK         SET AVMEME=FORSTK
        000
439 E33D9 146               C=DAT0 A               AS PART OF POLL'S ENTRY
```

```
440 E33DC 1B00          D0=(5) =AVMEME      CONDITIONS
          000
441 E33E3 144           DAT0=C A
442
443 E33E6 8F00          GOSBVL =POLL        ISSUE POLL
          000
444 E33ED 21            CON(2) =pENTER
445 E33EF 4A8           GOC    ERR          CHECK FOR ERROR FIRST
446
447 E33F2 831           ?XM=0               IF XM=0 THEN HANDLED
448 E33F5 41            GOYES ENT1
449 E33F7 8E00          GOSUBL =GETFP       POLL WAS NOT HANDLED,
          00
450 E33FD 07            C=RSTK              RECOVER UPDATED DATA STK PTR
451 E33FF 137           CD1EX
452 E3402 D2            C=0    A            RTN FALSE FLAG, SO "HPIL
453 E3404 145           DAT1=C A            ERROR" MSG WILL BE GIVEN
454 E3407 03            RTNCC
455
456 E3409 1F00 ENT1     D1=(5) =AVMEME      END OF DATA RECEIVED
          000
457 E3410 1B00          D0=(5) =FORSTK      START OF DATA RECEIVED
          000
458 E3417 AF0           A=0    W
459 E341A 147           C=DAT1 A
460 E341D 142           A=DAT0 A
461 E3420 130           D0=A                 SOURCE TO D0
462 E3423 EA            A=A-C  A            #NIBS RECEIVED
463 E3425 81C           ASRB                NIBS TO BYTES
464 E3428 D8            B=A    A            COPY OF # BYTES IN B
465 E342A 8A8           ?A=0   A            ANY RECEIVED?
466 E342D 93            GOYES  ENT2         NO
467 E342F 07            C=RSTK              RECOVER FORTH DATA STK PTR
468 E3431 06            RSTK=C              AND PUSH RIGHT BACK ON
469 E3433 135           D1=C
470 E3436 1C4           D1=D1-  5           point to max bytes to be
471           *                             transferred to FORTH
472 E3439 147           C=DAT1  A           read max bytes
473 E343C 8BA           ?C>=A   A           did we get less than or equal max b
474 E343F 60            GOYES   ENT02       yes, so use byte count in B(A) and
475 E3441 D5            B=C     A
476 E3443 DA            A=C     A           else revise max byte count
477 E3445 174  ENT02    D1=D1+  5           now point to addr
478
479 E3448 147           C=DAT1 A            DESTINATION ADDR
480 E344B 135           D1=C                TO D1
481 E344E 181           D0=D0-  2            POINT TO REAL 1ST CHAR!
482 E3451 CC   ENT01    A=A-1  A            DECREMENT COUNT
483 E3453 421           GOC    ENT2          DONE
484 E3456 14E           C=DAT0  B            READ CHAR
485 E3459 14D           DAT1=C  B            STORE CHAR
486 E345C 181           D0=D0-  2            DECR. SOURCE
487 E345F 171           D1=D1+  2            INCR. DEST.
488 E3462 6EEF          GOTO    ENT01
489
```



```
490 E3466 07   ENT2     C=RSTK              RECOVER PTR TO DATA STK
491 E3468 134           D0=C
492 E346B D9            C=B    A            C(A)=# bytes received
493 E346D 184           D0=D0- 5
494 E3470 144           DAT0=C A            PUSH STRING LEN
495 E3473 D2            C=0    A
496 E3475 CE            C=C-1  A            TRUE FLAG
497 E3477 184           D0=D0- 5
498 E347A 144           DAT0=C A            PUSH TRUE FLAG
499 E347D 132           AD0EX               SAVE CURRENT DATA STK PTR IN A(A)
500 E3480 8E00          GOSUBL =GETFP       RESTORE FORTH POINTERS
          00
501 E3486 133           AD1EX               REAL DATA STK PTR TO D1
502 E3489 03            RTNCC
503             ****************************************
504             *
505             * F.AND     Special Purpose Assembler Word
506             *
507             ****************************************
508
509 E348B 0000 =F.AND   CON(5)  =DOCOL      :
          0
510 E3490 0000          CON(5)  =ZEQ        0=
          0
511 E3495 0000          CON(5)  =SWAP       SWAP
          0
512 E349A 0000          CON(5)  =ZEQ        0=
          0
513 E349F 0000          CON(5)  =OR         OR
          0
514 E34A4 0000          CON(5)  =ZEQ        0=
          0
515 E34A9 0000          CON(5)  =SEMI       ;
          0
```

```
                EJECT
*******************************************************
*
* <FIND>   ( addr1  addr2 -- cfa  b  tf)   (ok)
*                         -- ff            (bad)
*
* This routine will search through RAM and ROM headers
* to find a dictionary entry.  RAM is always searched
* first, and it is assumed that there is always at least
* one RAM dictionary entry.
* In searching ROM we use the jump table pointing to
* the linked list of words of the length we are trying
* to match.
*
* addr2:  points to the header field addr at which to
*         start search
*
* If a match is found it returns the cfa addr, the
* number of BYTES in the name, and a true flag.
*
* If a match is not found, it simply returns a false flag.
*
*
* (High Mem)
*  --------
*
*             <code field addr>
*             <name>
*  addr2--->  <# bytes in name>
*             <link to next word>
*
*                      <name>
*  addr1--->       <# bytes in name>
* (addr1 points to the word which usually has been set up at
*  the end of the dictionary by WORD)
* (addr2 depends on the current vocabulary -- usually it points
*  to FORTH)
*  --------
* (Low Mem)
*
*
*
*

E34AE 3B43 =FIND>  CON(5) =$FIND>
      E

E34B3 147  =$FIND> C=DAT1  A          READ ADDR2
E34B6 DA           A=C     A          COPY OF ADDR2 TO A(A)
E34B8 136          CD0EX              D0=ADDR2
E34BB 108          R0=C               SAVE I IN R0
E34BE 174          D1=D1+  5
E34C1 147          C=DAT1  A          READ ADDR1
E34C4 D7           D=C     A          SAVE COPY OF ADDR1 IN D
E34C6 137          CD1EX              D1=ADDR1
```



```
E34C9 109          R1=C               SAVE DATA STK IN R1
E34CC AC1          B=0     S          B(S) is our RAM/ROM flag
                   *                  when B(S)=0 we are searching
                   *                  RAM, B(S)=F when we are
                   *                  searching ROM

                   * FND000  RAM SEARCH
                   *
E34CF 20   FND000  P=      0
E34D1 D2           C=0     A          CLEAR FOR LENGTH ADD
E34D3 31F1         LC(2)   31         SET MASK, MAX NAME LENGTH
                   *                  IS 31 CHARS
E34D7 DA           A=C     A          MASK TO A(B)
E34D9 14E          C=DAT0  B          LENGTH FROM ADDR2
E34DC 10A          R2=C               SAVE RAW LENGTH
E34DF 0E66         A=A&C   B          LENGTH ALONE
E34E3 14F          C=DAT1  B          LENGTH FROM ADDR1
E34E6 962          ?A=C    B          CHARS THE SAME?
E34E9 41           GOYES   MATCH1     YES!
                   *
E34EB 184  FND001  D0=D0-  5          BACK UP TO LINK
E34EE 146          C=DAT0  A          READ LINK TO C(A)
E34F1 8AA          ?C=0    A          OUT OF RAM ENTRIES?
E34F4 42           GOYES   SETROM     YES! NOW DO ROM ENTRIES
E34F6 134          D0=C               SET NEW ADDR2
E34F9 65DF         GOTO    FND000

                   * THE FOLLOWING CODE ENSURES THAT WE DON'T FIND
                   * DICTIONARY ENTRIES THAT ARE SMUDGED (I.E., IN THE
                   * PROCESS OF BEING DEFINED


E34FD 3102 MATCH1  LC(2)   #20        0010 0000  SMUDGE BIT MASK
E3501 14A          A=DAT0  B          READ LENGTH FROM ADDR2
E3504 0E66         A=A&C   B          "AND" WITH MASK
E3508 14F          C=DAT1  B          GET LENGTH ALONE FROM ADDR1
E350B 968          ?A=0    B          NOT SMUDGED?
E350E E5           GOYES   MATCH      YES
E3510 D0           A=0     A
E3512 AEA          A=C     B          GET LENGTH ALONE TO A(A)
E3515 55D          GONC    FND001       INTERMEDIATE JUMP

                   *
E3518 D0   SETROM  A=0     A
E351A 14B          A=DAT1  B            WORD LENGTH
E351D 31C0         LC(2)   12           NONE GREATER THAN 12
E3521 9E6          ?A>C    B
E3524 53           GOYES   FAIL         BAD LENGTH!
E3526 D9           C=B     A            SAVE RINSTK
E3528 D8           B=A     A            COPY LENGTH
E352A C4           A=A+A   A            LENGTH*2
E352C C4           A=A+A   A            LENGTH*4
E352E C0           A=A+B   A            LENGTH*5
E3530 D5           B=C     A            RESTORE B
```

```
625 E3532 3400          LC(5)   =TABLE
          000
626 E3539 C2            C=C+A   A               FIND JUMP-TABLE ENTRY
627 E353B 134           D0=C                    POINT TO ENTRY
628 E353E 146           C=DAT0  A               GET ADDR2
629 E3541 134           D0=C
630 E3544 A4D           B=B-1   S               SET ROM FLAG
631 E3547 6420          GOTO    MATCH
632             *
633
634 E354B 184   FND01   D0=D0-  5               POINT TO LINK
635 E354E 146           C=DAT0  A               GET LINK
636 E3551 134           D0=C                    SET D0 TO NEXT ITEM
637 E3554 8AE           ?C#0    A               IS IT A REAL LINK?
638 E3557 51            GOYES   MATCH
639 E3559 D0    FAIL    A=0     A               FALSE FLAG
640 E355B 118           C=R0
641 E355E 134           D0=C                    RECOVER I
642 E3561 119           C=R1
643 E3564 135           D1=C                    RECOVER RTN STK
644 E3567 141           DAT1=A  A               PUSH FAIL FLAG
645 E356A 03            RTNCC
646
647 E356C 136   MATCH   CD0EX
648 E356F 06            RSTK=C                  SAVE ADDR2 ON CPU RTN STK
649 E3571 134           D0=C                    RESTORE ADDR2 TO D0
650 E3574 D2            C=0     A
651 E3576 14E           C=DAT0  B               READ RAW LENGTH
652 E3579 10A           R2=C
653
654 E357C 171   MAT00   D1=D1+  2               point addr1 to 1st char
655 E357F 161           D0=D0+  2               point addr2 to 1st char
656 E3582 14B           A=DAT1  B               char from addr1
657 E3585 A64           A=A+A   B               *2
658 E3588 14E           C=DAT0  B               char from addr2
659 E358B A66           C=C+C   B               *2 set carry if high bit set
660 E358E 4A1           GOC     LSTCHR          last char!
661 E3591 962           ?C=A    B               chars equal?
662 E3594 8E            GOYES   MAT00           yes, do another
663             *
664             * WE WERE MATCHING CHARS, NOW WE'VE FAILED. NEED TO
665             * DETERMINE WHETHER WE'RE SEARCHING RAM OR ROM
666             *
667 E3596 DB    FND1    C=D     A
668 E3598 135           D1=C                    RESTORE ADDR1
669 E359B 07            C=RSTK                  POP ADDR2
670 E359D 136           CD0EX                   RESET D0 TO ADDR2
671 E35A0 94D           ?B#0    S               DOING ROM?
672 E35A3 8A            GOYES   FND01           YES
673 E35A5 654F          GOTO    FND001          NO, DOING ROM
674
675 E35A9 966   LSTCHR  ?C#A    B               chars unequal?
676 E35AC AE            GOYES   FND1            YES try another
677
678 E35AE 161   SUCESS  D0=D0+  2               point to cfa
```



```
679 E35B1 136           CD0EX                   GET IT IN C[A]
680             *
681 E35B4 111   SUC01   A=R1
682 E35B7 131           D1=A                    RESTORE DATA STACK PTR
683 E35BA 145           DAT1=C  A               WRITE CFA TO STACK
684 E35BD 1C4           D1=D1-  5
685 E35C0 11A           C=R2                    length of name
686 E35C3 145           DAT1=C  A               PUSH LENGTH
687 E35C6 1C4           D1=D1-  5
688 E35C9 D0            A=0     A
689 E35CB CC            A=A-1   A               TRUE=ALL ONE'S
690 E35CD 141           DAT1=A  A               PUSH FLAG
691 E35D0 07            C=RSTK                  throw away addr2
692 E35D2 118           C=R0
693 E35D5 134           D0=C                    RESTORE I TO D0
694 E35D8 03            RTNCC
695
696
```

```
                        EJECT
                *********************************
                *
                * 2OVER (d1 d2 --- d1 d2 d1)
                *
                *********************************

E35DA A533E     =!ENTER CON(5)  =!ENTER
E35DF 58        =!OVER2 CON(2)  #85             2OVER
E35E1 23F46554  =2OVE   NIBASC  \2OVE\
E35E9 2D        =OVER   CON(2)  \R\+#80
E35EB 0F53E     =$OVER2 CON(5)  =$OVER2

E35F0 179       =$OVER2  D1=D1+  10              POINT TO D1
E35F3 15F9              C=DAT1  10              READ D1
E35F7 1C9               D1=D1-  10
E35FA 1C9               D1=D1-  10
E35FD 15D9              DAT1=C  10              WRITE D1 AGAIN
E3601 03                RTNCC
```



```
                        EJECT
                ************************************************************
                *
                * NMOVE (addr1 addr2 u --- ) NOTE: THIS RTNE REPLACES
                *       CMOVE IN THE FORTH79 STANDARD, IT MOVES
                *       NIBBLES RATHER THAN BYTES.
                *
                * If u=0 do nothing, otherwise treat u as an unsigned
                * number.
                * move n nibbles from addr1 to addr2, start at addr1 and
                * move toward higher memory.
                *
                * Copying is done in blocks of 16 nibbles until there are
                * less than 16 left to do, then one at a time.
                *
                * It uses 1 additional sublevel, R1,R2, C, A, D0, D1, P
                *
                ************************************************************
E3603 FD53E     =!OVER2 CON(5)  =!OVER2
E3608 58        =!NMOVE CON(2)  #85             NMOVE
E360A E4D4F465  =NMOV   NIBASC  \NMOV\
E3612 5C                CON(2)  \E\+#80
E3614 9163E     =NMOVE  CON(5)  =$NMOVE

E3619 AF0       =$NMOVE A=0     W               CLEAR FOR SHIFT IN MOVSET
E361C 143               A=DAT1  A               GET #OF NIBS
E361F 7340      NMOVE0  GOSUB   MOVSET          A[A]=#WORDS;A[S]=#NIBS+
                *                               D1=FROM D0=TO
                * If zero nibs to move, MOVSET pops a return, cleans up
                * data stack and whole word is done.
                *
E3623 8A8       =NMOVE1 ?A=0    A               NO MORE WORDS?
E3626 61                GOYES   NMOVE2
E3628 1577              C=DAT1  W               GET A WORD
E362C 1547              DAT0=C  W               WRITE ONE
E3630 17F               D1=D1+  16              INCREMENT FROM (ADDR1)
E3633 16F               D0=D0+  16                  INCREMENT ADDR2
E3636 CC                A=A-1   A                   DECREMENT COUNT
E3638 6AEF              GOTO    NMOVE1
E363C 810       NMOVE2  ASLC                    GET NIBS INTO A[A]
E363F 8A8       NMOVE3  ?A=0    A
E3642 61                GOYES   CM3
E3644 15F0              C=DAT1  1
E3648 15C0              DAT0=C  1
E364C 170               D1=D1+  1
E364F 160               D0=D0+  1
E3652 CC                A=A-1   A
E3654 6AEF              GOTO    NMOVE3
E3658 119       =CM3    C=R1
E365B 134               D0=C                    RECOVER I
E365E 11A               C=R2
E3661 135               D1=C                    RECOVER DATA STACK
```

```
768 E3664 03          RTNCC
769
770           *
771           * Set up counter for number of words (16-nibble blocks) plus
772           * odd number of nibbles; set D0 & D1 to the target and destinatio
773           *
774 E3666 8AC  =MOVSET ?A#0    A             ANY TO DO?
775 E3669 90          GOYES   MVCN1         YES
776 E366B 17E         D1=D1+  15            NO, THROW AWAY 2 ADDRS
777 E366E 07          C=RSTK                DO A 2 LEVEL RETURN
778 E3670 03          RTNCC
779 E3672 814  MVCN1  ASRC                  DIVIDE BY 16,REMAINDER IN A[S]
780 E3675 174         D1=D1+  5
781 E3678 147         C=DAT1  A             POP ADDR 2
782 E367B 136         CD0EX                 D0=ADDR2
783 E367E 109         R1=C                  SAVE I IN R1
784 E3681 174         D1=D1+  5
785 E3684 147         C=DAT1  A             POP ADDR1
786 E3687 174         D1=D1+  5             NOTHING RETURNED
787 E368A 137         CD1EX                 D1=ADDR1
788 E368D 10A         R2=C                  SAVE DATA STK IN R2
789 E3690 03          RTNCC
```



```
790           EJECT
791          ************************************************************
792          *
793          * 2DROP  (d --- ) drop double number from top of data stack
794          *
795          ************************************************************
796
797 E3692 8063          CON(5)  =!NMOVE
         E
798 E3697 58   =!2DROP  CON(2)  #85           2DROP
799 E3699 2344         NIBASC  \2DRO\
         25F4
800 E36A1 0D           CON(2)  \P\+#80
801 E36A3 8A63 =2DROP  CON(5)  =$2DROP
         E
802
803 E36A8 179  =$2DROP D1=D1+ 10
804 E36AB 03          RTNCC
```

```
805                     EJECT
806                 ********************************************************
807                 *
808                 * DIGIT  (c n1 --- n2 flag )  ok
809                 *        (c n1 --- flag    )  bad
810                 *
811                 * converts the ASCII char c (using base n1) to its binary
812                 * equivalent n2, accompanied by a true flag. If conversion
813                 * is invalid, leaves only a false flag.
814                 *
815                 ********************************************************
816 E36AD 7963          CON(5)  =!2DROP
        E
817 E36B2 58    =!DIGIT CON(2)  #85             DIGIT
818 E36B4 4494        NIBASC  \DIGI\
        7494
819 E36BC 4D          CON(2)  \I\+#80
820 E36BE 3C63 =DIGIT  CON(5)  =$DIGIT
        E
821
822 E36C3 174   =$DIGIT D1=D1+  5
823 E36C6 AF0           A=0     W
824 E36C9 14B           A=DAT1  B               READ CHAR
825 E36CC 20            P=      0
826 E36CE 31A3          LC(2)   #3A             ':'
827 E36D2 B6E           C=A-C   B               CHAR <= 9?
828 E36D5 413           GOC     DIG30           YES!
829 E36D8 3114          LC(2)   #41             'A'
830 E36DC B6E           C=A-C   B               CHAR < A?
831 E36DF 413           GOC     DIG40           YES, THEN ERROR
832 E36E2 3173          LC(2)   #37
833 E36E6 B6A           A=A-C   B               MAKE 'A'=10
834 E36E9 1C4   DIG20   D1=D1-  5
835 E36EC 14F           C=DAT1  B               READ BASE
836 E36EF 174           D1=D1+  5
837 E36F2 B6E           C=A-C   B               CHAR > BASE?
838 E36F5 5B1           GONC    DIG40           YES, ERROR
839 E36F8 141           DAT1=A  A               WRITE CHAR
840 E36FB 1C4           D1=D1-  5
841 E36FE D2            C=0     A
842 E3700 CE            C=C-1   A               TRUE=ALL 1'S
843 E3702 145   DIG25    DAT1=C  A               PUSH FLAG
844 E3705 03            RTNCC
845 E3707 3103 DIG30    LC(2)   #30
846 E370B B6A           A=A-C   B               CHAR < 0?
847 E370E 5AD           GONC    DIG20           NO, CHAR OKAY
848 E3711 D2    DIG40   C=0     A               FALSE FLAG
849 E3713 6EEF          GOTO    DIG25
```



```
850                     EJECT
851                 ********************************************************
852                 *
853                 * 2SWAP  (d1 d2 --- d2 d1)  swap order of 2 dbl #s
854                 *
855                 ********************************************************
856
857 E3717 2B63         CON(5)  =!DIGIT
        E
858 E371C 58    =!SWAP2 CON(2)  #85             SWAP2
859 E371E 2335      NIBASC  \2SWA\
        7514
860 E3726 0D          CON(2)  \P\+#80
861 E3728 D273 =SWAP2  CON(5)  =$SWAP2
        E
862
863 E372D 15B9 =$SWAP2 A=DAT1  10               POP D2
864 E3731 179         D1=D1+  10
865 E3734 15F9        C=DAT1  10               POP D1
866 E3738 1599        DAT1=A  10               PUSH D2
867 E373C 1C9         D1=D1-  10
868 E373F 15D9        DAT1=C  10               PUSH D1
869 E3743 03          RTNCC
```

```
870                     EJECT
871                 ****************************************************************
872                 *
873                 * COUNT  (addr --- addr+1 n) leave on data stack the address, add
874                 *        and the char count of text beginning at addr.  1st byte
875                 *        must contain the char count, n.
876                 ****************************************************************
877
878 E3745 C173         CON(5)  =!SWAP2
        E
879 E374A 58    =!COUNT CON(2)  #85             COUNT
880 E374C 34F4         NIBASC  \COUN\
        55E4
881 E3754 4D           CON(2)  \T\+#80
882 E3756 B573 =COUNT  CON(5)  =$COUNT
        E
883
884 E375B 143   =$COUNT A=DAT1  A               POP ADDR
885 E375E 132          ADOEX                   D0=ADDR, A=I
886 E3761 D2           C=0     A
887 E3763 14E          C=DAT0  B               GET CHAR CNT AT ADDR
888 E3766 161          D0=D0+  2               ADDR+1
889 E3769 132          ADOEX                   D0=I, A=ADDR+1
890 E376C 141          DAT1=A  A               PUSH ADDR+1
891 E376F 1C4          D1=D1-  5
892 E3772 145          DAT1=C  A               PUSH COUNT
893 E3775 03           RTNCC
```



```
894                     EJECT
895                 ****************************************************
896                 *
897                 * WIDTH  ( --- addr)  addr of variable with max
898                 *       name length.
899                 *
900                 ****************************************************
901
902 E3777 A473         CON(5)  =!COUNT
        E
903 E377C 58    =!WIDTH CON(2)  #85             WIDTH
904 E377E 7594         NIBASC  \WIDT\
        4445
905 E3786 8C           CON(2)  \H\+#80
906 E3788 D873 =WIDTH  CON(5)  =$WIDTH
        E
907
908 E378D 8E00 =$WIDTH GOSUBL  =DOUSE           MAX NAME LENGTH
        00
909 E3793 34BF         CON(5)   (=oWIDTH)
        2
910
911                 ****************************************
912                 *
913                 * FENCE  ( --- addr) addr of variable with
914                 *       addr of protected dictionary entries
915                 *
916                 ****************************************
917 E3798 C773         CON(5)  =!WIDTH
        E
918 E379D 58    =!FENCE CON(2)  #85             FENCE
919 E379F 6454         NIBASC  \FENC\
        E434
920 E37A7 5C           CON(2)  \E\+#80
921 E37A9 EA73 =FENCE  CON(5)  =$FENCE
        E
922
923 E37AE 8E00 =$FENCE GOSUBL  =DOUSE
        00
924 E37B4 E8BF         CON(5)   (=oFENCE)
        2
```

```
925                     EJECT
926                 ********************************************
927                 *
928                 * VOC-L  ( --- addr)  addr of variable with
929                 *       addr of the parent-vocabulary-field of
930                 *       the most recently created vocabulary.
931                 *
932                 ********************************************
933
934 E37B9 EB73  =VOC-L  CON(5)  =$VOC-L
       E
935 E37BE 8E00  =$VOC-L GOSUBL  =DOUSE
       00
936 E37C4 43BF         CON(5)   (=oVOC-L)
       2
937
938                 ****************************************
939                 *
940                 * OKFLG  ( --- ) addr of variable with
941                 *       flag:  0=show OK, 1=suppress OK
942                 *
943                 ****************************************
944
945 E37C9 D973         CON(5)  =!FENCE
       E
946 E37CE 58    =!OKFLG CON(2)  #85             OKFLG
947 E37D0 F4B4         NIBASC  \OKFL\
       64C4
948 E37D8 7C           CON(2)  \G\+#80
949 E37DA FD73  =OKFLG  CON(5)  =$OKFLG
       E
950
951 E37DF 8E00  =$OKFLG GOSUBL  =DOUSE
       00
952 E37E5 D4BF         CON(5)   (=oOKFLG)
       2
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER


```
953                     EJECT
954
955                 ****************************************
956                 *
957                 * ONERR
958                 *
959                 * FLAG TO SAY IF ON ERROR IS SET -- IF
960                 * SO IT HAS THE EXECUTION ADDRESS OF THE
961                 * ROUTINE TO GO TO.
962                 *
963                 * ONERR is zeroed every time you enter FORTH,
964                 * and everytime you do an INIT1
965                 *
966                 ****************************************
967 E37EA EC73         CON(5)  =!OKFLG
       E
968 E37EF 58    =!ONERR CON(2)  #85
969 E37F1 F4E4         NIBASC  \ONER\
       5425
970 E37F9 2D           CON(2)  \R\+#80
971 E37FB 0083  =ONERR  CON(5)  =$ONERR
       E
972 E3800 8E00  =$ONERR GOSUBL  =DOUSE
       00
973 E3806 6BBF         CON(5)  (=oONERR)
       2
974
975                 ********************************************
976                 *
977                 * STATE  ( --- addr) addr of variable with
978                 *       flag showing current state: execute (0)
979                 *       or compile (C0)
980                 *
981                 ********************************************
982
983 E380B FE73         CON(5)  =!ONERR
       E
984 E3810 58    =!STATE CON(2)  #85             STATE
985 E3812 3545         NIBASC  \STAT\
       1445
986 E381A 5C           CON(2)  \E\+#80
987 E381C 1283  =STATE  CON(5)  =$STATE
       E
988
989 E3821 8E00  =$STATE GOSUBL  =DOUSE
       00
990 E3827 07BF         CON(5)   (=oSTATE)
       2
991
992
993                 ****************************************
994                 *
995                 * FIRST  ( --- addr)  addr of variable with
996                 *       addr of 1st disc buffer
997                 *
```

```
 998            ********************************************
 999
1000 E382C 0183         CON(5)  =!STATE
           E
1001 E3831 58   =!FIRST CON(2) #85            FIRST
1002 E3833 6494         NIBASC \FIRS\
           2535
1003 E383B 4D           CON(2) \T\+#80
1004 E383D 2483 =FIRST  CON(5)  =$FIRST
           E
1005
1006 E3842 8E00 =$FIRST GOSUBL =DOUSE
           00
1007 E3848 A2BF         CON(5) (=oFIRST)
           2
1008
1009
1010            ********************************************
1011            *
1012            * LIMIT  ( --- addr)  addr of variable with
1013            *       addr of 1st byte beyond disc buffer
1014            *       area
1015            *
1016            ********************************************
1017
1018 E384D 1383         CON(5)  =!FIRST
           E
1019 E3852 58   =!LIMIT CON(2) #85            LIMIT
1020 E3854 C494         NIBASC \LIMI\
           D494
1021 E385C 4D           CON(2) \T\+#80
1022 E385E 3683 =LIMIT  CON(5)  =$LIMIT
           E
1023
1024 E3863 8E00 =$LIMIT GOSUBL =DOUSE
           00
1025 E3869 F2BF         CON(5) (=oLIMIT)
           2
1026
1027            ********************************************
1028            *
1029            * LINE#  ( --- addr)  addr of variable with
1030            *       line# associated with current buffer
1031            *       (usually PREV buffer)
1032            *
1033            ********************************************
1034
1035 E386E 2583         CON(5)  =!LIMIT
           E
1036 E3873 58   =!LINE# CON(2) #85            LINE#
1037 E3875 C494         NIBASC \LINE\
           E454
1038 E387D 3A           CON(2) \*\+#80
1039 E387F 4883 =LINE#  CON(5)  =$LINE#
           E
```



```
1040
1041 E3884 8E00 =$LINE# GOSUBL  =DOUSE
           00
1042 E388A D9BF         CON(5) (=oLINE#)
           2
```

```
1043                    EJECT
1044                ***********************************************
1045                *
1046                * LEAVE  ( --- ) force termination of a LOOP or +LOOP
1047                *         by setting loop limit = to current index value
1048                *
1049                * IMMEDIATE, COMPILE
1050                *
1051                ***************************************************
1052
1053 E388F 3783          CON(5)  =!LINE#
          E
1054 E3894 5C    =!LEAVE CON(2)  #C5             LEAVE
1055 E3896 C454          NIBASC  \LEAV\
          1465
1056 E389E 5C            CON(2)  \E\+#80
1057 E38A0 0000 =LEV     CON(5)  =DOCOL          :
          0
1058 E38A5 0000          CON(5)  =COMPL          compile next word into dictionary
          0
1059 E38AA 0000          CON(5)  =R>             RECOVER LOOP INDEX
          0
1060 E38AF 0000          CON(5)  =COMPL          compile next word into dictionary
          0
1061 E38B4 0000          CON(5)  =R>             RECOVER LOOP LIMIT
          0
1062 E38B9 0000          CON(5)  =COMPL          compile next word into dictionary
          0
1063 E38BE 3A63          CON(5)  =2DROP          THROW AWAY LOOP LIMIT AND INDEX
          E
1064 E38C3 0000          CON(5)  =COMPL          compile next word into dictionary
          0
1065 E38C8 0000          CON(5)  =BRNCH
          0
1066 E38CD 0193          CON(5)  =DOLEV          HANDLE SETTING STACK WITH LEAVES
          E
1067 E38D2 22E3          CON(5)  =ABORTW         GIVE "no DO before LEAVE"
          E
1068 E38D7 B3            CON(2) =eNODO          MESSAGE
1069 E38D9 0000          CON(5) =HERE           LEAVE ITEMS AT, HERE GIVES ADDR TO
          0
1070 E38DE 0000          CON(5) =OVER
          0
1071 E38E3 0000          CON(5) =STORE          ADDR PART OF LEAVE STORED
          0
1072 E38E8 0000          CON(5) =FIVE-          DROP ADJ STK POINTER TO STORE LEAVE
          0
1073 E38ED 0000          CON(5) =LIT
          0
1074 E38F2 6000          CON(5) 6
          0
1075 E38F7 0000          CON(5) =SWAP
          0
1076 E38FC 0000          CON(5) =STORE          STORE 6
          0
```



```
1077 E3901 0000          CON(5) =ZERO           PUT 0 IN DICTIONARY TO SAVE PLACE
          0
1078 E3906 0000          CON(5) =COMMA          FOR BRNCH TO BE FILLED IN  AT LOOP,
          0
1079 E390B 0000          CON(5) =SEMI           DONE!
          0
1080
1081 E3910 5193 DOLEV    CON(5) $DOLEV
          E
1082
1083 E3915 8E00 $DOLEV   GOSUBL =SAVEFP         SAVE FORTH ENVIRONMENT
          00
1084 E391B 1B48          D0=(5) =oCSP           RECOVER VALUE OF STK PTR WHEN
          BF2
1085 E3922 146           C=DAT0 A               COMPILATION STARTED
1086 E3925 D7            D=C    A               SAVE IN D FOR LATER TESTS
1087 E3927 3430          LC(5)  3               LOOKING FOR 3 LEFT BY 'DO' CONSTRUC
          000
1088 E392E DA            A=C    A               PUT 3 IN A
1089 E3930 137           CD1EX
1090 E3933 134           D0=C                   SET D0 TO CURRENT TOS-10 (ADDING 2
1091 E3936 189           D0=D0- 10              WE WILL MOVE PAIRS AS WE GO
1092 E3939 135           D1=C                   RESTORE TOS TO D1
1093
1094 E393C 179  DOLEV0   D1=D1+ 10              CHECK TO SEE THAT THERE ARE AT LEAS
1095 E393F 137           CD1EX                  2 ITEMS ON STACK BEFORE SHIFTING ST
1096 E3942 8B7           ?C>D   A               TOS LESS 2 ITEMS BEYOND MARKER?
1097 E3945 34            GOYES  BAD             YES, DO BAD EXIT
1098 E3947 135           D1=C
1099 E394A 1C9           D1=D1- 10              RESTORE TOS
1100
1101 E394D 147           C=DAT1 A               READ FLAG FROM PAIR
1102 E3950 8A6           ?A#C   A               IS THIS A 'DO' FLAG?
1103 E3953 40            GOYES  DOLEV1          NO,
1104 E3955 D3            D=0    A               YES, MAKE D FLAG THIS
1105 E3957 144  DOLEV1   DAT0=C A               WRITE CONDITIONAL FLAG UP 2
1106 E395A 174           D1=D1+ 5               POINT TO CONDITIONAL ADDR
1107 E395D 164           D0=D0+ 5
1108 E3960 147           C=DAT1 A               READ ADDR
1109 E3963 144           DAT0=C A               WRITE CONDITIONAL ADDR UP 2
1110 E3966 174           D1=D1+ 5
1111 E3969 164           D0=D0+ 5               SET PTRS FOR NEXT PAIR
1112 E396C 8AF           ?D#0   A               ARE WE STILL LOOKING FOR A 'DO' FLA
1113 E396F DC            GOYES  DOLEV0          YES TRY AGAIN
1114
1115 E3971 1C4           D1=D1- 5               POINT TO ADDR PART OF PAIR JUST MOV
1116 E3974 133           AD1EX                  PASS THIS BACK TO REST OF LEAVE
1117 E3977 8E00          GOSUBL =GETFP          A(A) IS NOT DISTURBED BY THIS
          00
1118 E397D 1CE           D1=D1- 15              MUST ALLOW FOR 2 ITEMS ADDED AND
1119 E3980 141           DAT1=A A               PUSH ADDR TO PUT LEAVE PAIR
1120 E3983 D0            A=0    A               FALSE FLAG FOR ABORT
1121 E3985 5C0           GONC   BAD0            B.E.T.
1122
1123 E3988 8E00 BAD      GOSUBL =GETFP          RESTORE FORTH ENVIRONMENT
```

```
            00
1124 E398E D0           A=0    A            TRUE FLAG FOR  ABORT
1125 E3990 CC           A=A-1  A
1126 E3992 1C4  BAD0    D1=D1- 5
1127 E3995 141          DAT1=A A             PUSH FLAG
1128 E3998 03           RTNCC
```



```
1129                    EJECT
1130                *****************************************************
1131                *
1132                * DOES> ( - ) This is an immediate word used in the
1133                * construction of new defining words.
1134                * Since it is immediate, its run-time is the compile-time
1135                * of the new defining word.
1136                * The action of DOES> is to enclose
1137                *       CON(5)  =<;CODE>
1138                *       GOSBVL =DODOES
1139                * in the definition of the defining word.
1140                *
1141                *************************************************
1142
1143 E399A 4983             CON(5)  =!LEAVE
            E
1144 E399F 5C   =!DOES> CON(2)  #C5             DOES>
1145 E39A1 44F4             NIBASC  \DOES\
            5435
1146 E39A9 EB             CON(2)  \>\+#80
1147
1148 E39AB 0000 =DOES>  CON(5) =DOCOL
            0
1149 E39B0 0000             CON(5)  =QCSP       ?CSP   give error if stack is not
            0
1150                *                           same as when compile started
1151 E39B5 0000             CON(5)  =COMPL      COMPILE
            0
1152 E39BA 0000             CON(5)  =<;COD      <;CODE>
            0
1153 E39BF 0000             CON(5)  =LIT        insert
            0
1154 E39C4 8F00             CON(5) #F8          op code for GOSBVL
            0
1155 E39C9 0000             CON(5)  =C,         C,
            0
1156 E39CE 0000             CON(5)  =LIT
            0
1157 E39D3 0000             CON(5) =DODOES      dodoes
            0
1158 E39D8 0000             CON(5)  =COMMA      ,
            0
1159 E39DD 0000             CON(5)  =SEMI       ;
            0
```

```
1160                    EJECT
1161                ************************************************
1162                *
1163                * BEGIN   compile time: ( - addr 1 )    immediate
1164                * no run-time action
1165                *
1166                ************************************************
1167
1168 E39E2 F993           CON(5) =!DOES>
         E
1169 E39E7 5C     =!BEGIN CON(2) #C5            BEGIN
1170 E39E9 2454           NIBASC \BEGI\
         7494
1171 E39F1 EC           CON(2) \N\+#80
1172
1173 E39F3 0000   =BEGIN  CON(5) =DOCOL
         0
1174 E39F8 0EB3           CON(5)  =?COMP        error if we're no compiling
         E
1175 E39FD 0000           CON(5)  =HERE         addr to branch to later
         0
1176 E3A02 0000           CON(5)  =ONE          1 BEGIN's flag
         0
1177 E3A07 0000           CON(5)  =SEMI         ;
         0
1178
1179                ************************************************
1180                *
1181                * AGAIN   compile time: ( addr 1 - )   immediate
1182                * compiles BRANCH ( - )
1183                *
1184                *  MADE HEADERLESS 10/10/83
1185                ************************************************
1186
1187 E3A0C 0000 =AGAIN  CON(5) =DOCOL
         0
1188 E3A11 0000           CON(5)  =ONE          1   look for BEGIN's 1
         0
1189 E3A16 0000           CON(5)  =?PAIR        error if not there
         0
1190 E3A1B 0000           CON(5)  =COMPL        COMPILE  unconditional
         0
1191 E3A20 0000           CON(5)  =BRNCH        BRANCH
         0
1192 E3A25 0000           CON(5)  =HERE         get offset; old HERE - new HERE
         0
1193 E3A2A 0000           CON(5)  =MINUS        -
         0
1194 E3A2F 0000           CON(5)  =COMMA        ,
         0
1195 E3A34 0000           CON(5)  =SEMI         ;
         0
```



```
1196                    EJECT
1197                ************************************************
1198                *
1199                * UNTIL   compile time: ( addr 1 - )   immediate
1200                * compiles 0BRANCH ( f - )
1201                * with jump back to addr of BEGIN which it left
1202                * on the stack
1203                *
1204                ************************************************
1205
1206 E3A39 7E93           CON(5)  =!BEGIN
         E
1207 E3A3E 5C     =!UNTIL CON(2) #C5            UNTIL
1208 E3A40 55E4           NIBASC \UNTI\
         4594
1209 E3A48 CC           CON(2) \L\+#80
1210
1211 E3A4A 0000   =UNTIL  CON(5) =DOCOL
         0
1212 E3A4F 0EB3           CON(5)  =?COMP        ensure compile mode
         E
1213 E3A54 0000           CON(5)  =ONE          1
         0
1214 E3A59 0000           CON(5)  =?PAIR        error if BEGIN flag absent
         0
1215 E3A5E 0000           CON(5)  =COMPL        COMPILE conditional
         0
1216 E3A63 0000           CON(5)  =ZBRNH        0BRANCH
         0
1217 E3A68 0000           CON(5)  =HERE         old HERE - new HERE
         0
1218 E3A6D 0000           CON(5)  =MINUS        get offset back to BEGIN
         0
1219 E3A72 0000           CON(5)  =COMMA        store into dictionary as
         0
1220 E3A77 0000           CON(5)  =SEMI          offset for the ZBRNH
         0
1221
1222                ************************************************
1223                *
1224                * WHILE   compile time: ( - addr 4 )    immediate
1225                * compiles 0BRANCH ( f - )
1226                * with offset to be filled in later by REPEAT
1227                *
1228                ************************************************
1229
1230 E3A7C E3A3           CON(5)  =!UNTIL
         E
1231 E3A81 5C     =!WHILE CON(2) #C5            WHILE
1232 E3A83 7584           NIBASC \WHIL\
         94C4
1233 E3A8B 5C           CON(2) \E\+#80
1234
1235 E3A8D 0000   =WHILE  CON(5) =DOCOL
         0
```

```
1236 E3A92 0000          CON(5)  =IF          same action as IF, compiling
           0
1237             *                            ZBRNH and allotting space for
1238             *                            offset to be filled in later
1239 E3A97 0000          CON(5)  =TWO+        except we leave 4 instead of 2
           0
1240 E3A9C 0000          CON(5)  =SEMI        as syntax-check code for REPEAT
           0
```



```
1241                     EJECT
1242             ********************************************************
1243             *
1244             * +LOOP  (n --- )  Add the signed increment to the loop
1245             *                  index and compare the new total to the loop
1246             *                  limit.
1247             *
1248             ********************************************************************
1249
1250 E3AA1 18A3          CON(5)  =!WHILE
           E
1251 E3AA6 5C    =!PLOOP CON(2)  #C5          +LOOP
1252 E3AA8 B2C4          NIBASC  \+LOO\
           F4F4
1253 E3AB0 0D            CON(2)  \P\+#80
1254
1255 E3AB2 0000  =PLOOP  CON(5)  =DOCOL       :
           0
1256 E3AB7 0EB3          CON(5)  =?COMP       ensure compile mode
           E
1257 E3ABC 0000          CON(5)   =THREE      3
           0
1258 E3AC1 0000          CON(5)   =?PAIR      error if DO flag is absent
           0
1259 E3AC6 0000          CON(5)   =COMPL      COMPILE
           0
1260 E3ACB 18F3          CON(5)   =<+LP>      <+LOOP>
           E
1261 E3AD0 0000          CON(5)   =HERE       old HERE - new HERE
           0
1262 E3AD5 0000          CON(5)   =MINUS      -
           0
1263 E3ADA 0000          CON(5)   =COMMA      ,
           0
1264 E3ADF 0000          CON(5)   =HERE       INPUT FOR SETLEV
           0
1265 E3AE4 0000          CON(5)   =SETLEV     SET UP LEAVE BRANCHES IF ANY
           0
1266 E3AE9 0000          CON(5)   =SEMI       ;
           0
```

```
1267                    EJECT
1268              ************************************************************
1269              *
1270              * CMOVE  (addr1 addr2 n --- )  Move u BYTES starting at addr1
1271              *        to addr2. if u=0 do nothing.
1272              *
1273              ************************************************************
1274
1275 E3AEE 6AA3          CON(5)  =!PLOOP
          E
1276 E3AF3 58    =!CMOVE CON(2)  #85           CMOVE
1277 E3AF5 34D4          NIBASC  \CMOV\
          F465
1278 E3AFD 5C            CON(2)  \E\+#80
1279
1280 E3AFF 40B3 =CMOVE   CON(5)  $CMOVE
          E
1281 E3B04 AF0  $CMOVE   A=0     W            CLEAR FOR SHIFT IN MOVSET
1282 E3B07 143           A=DAT1  A            GET #OF BYTES TO MOVE
1283 E3B0A C4            A=A+A   A            MAKE NIBS
1284 E3B0C 621B          GOTO    NMOVE0
```



```
1285                    EJECT
1286              ************************************************
1287              *
1288              * QUERY  ( --- )  Accepts input from the terminal.
1289              *        EXPECT calls mainframe rtne CHEDIT to get
1290              *        user input so, the input is not limited to
1291              *        a certain char count here.  The input is
1292              *        moved to the terminal input buffer.
1293              *
1294              *        Also, as per FORTH 83 Standard, the user
1295              *        variable #TIB is set to reflect the number
1296              *        of characters put into the Terminal Input
1297              *        Buffer.  #TIB is set from SPAN which is
1298              *        set by EXPECT96
1299              *
1300              ************************************************
1301
1302 E3B10 3FA3          CON(5)  =!CMOVE
          E
1303 E3B15 58    =!QUERY CON(2)  #85           QUERY
1304 E3B17 1555          NIBASC  \QUER\
          5425
1305 E3B1F 9D            CON(2)  \Y\+#80
1306
1307 E3B21 0000  =QUERY  CON(5)  =DOCOL        :
          0
1308 E3B26 0000          CON(5)   =TIB                 addr of TIB
          0
1309 E3B2B 0000          CON(5)   =EXPCT       EXPECT  for EXPECT to put input
          0
1310 E3B30 0000          CON(5)   =ZERO        0       set pointer into
          0
1311 E3B35 0000          CON(5)   =IN          IN      input at 0 chars
          0
1312 E3B3A 0000          CON(5)   =STORE       !
          0
1313 E3B3F 0000          CON(5)   =SPAN        SPAN    get # chars read
          0
1314 E3B44 0000          CON(5)   =AT          @       by EXPECT96 and
          0
1315 E3B49 0000          CON(5)   =#TIB        #TIB    put in #TIB
          0
1316 E3B4E 0000          CON(5)   =STORE       !
          0
1317 E3B53 0000          CON(5)   =SEMI        ;
          0
```

```
1318                    EJECT
1319                ******************************************************
1320                *
1321                * ALLOT  (n --- )  Add n bytes to the parameter field
1322                *        of the most recently defined word.
1323                *
1324                *        Although its not standard FORTH we have
1325                * changed ALLOT to try and protect users from ruining
1326                * their dictionary and data stack.  Before alloting
1327                * any additional space in the dictionary we make sure
1328                * that the current DP setting allows for the entire
1329                * data stack (40 entries or 200 nibs) and the PAD area
1330                * (90 nibs) plus the # nibs to be alloted.
1331                *
1332                ******************************************************
1333
1334 E3B58 51B3E            CON(5)  =!QUERY
1335 E3B5D 58       =!ALOTB CON(2)  #85             ALLOT
1336 E3B5F 14C4C4F4         NIBASC  \ALLO\
1337 E3B67 4D               CON(2)  \T\+#80
1338
1339 E3B69 00000 =ALOTB    CON(5)  =DOCOL          :
1340 E3B6E 00000           CON(5)  =TWO*           TURN BYTES TO NIBS
1341 E3B73 00000           CON(5)  =ALLOT          DO ALLOT
1342 E3B78 00000           CON(5)  =SEMI           ;
1343
1344                *
1345                * DEFAULT ALLOT ROUTINE
1346                *
1347
1348 E3B7D 00000 =IALLOT   CON(5)  =DOCOL          :
1349 E3B82 00000           CON(5)  =DUP
1350 E3B87 00000           CON(5)  =>R
1351 E3B8C 00000           CON(5)   =DP             DP
1352 E3B91 00000           CON(5)  =DUP            DUP copy of DP
1353 E3B96 00000           CON(5)  =AT             @   current value of DP
1354 E3B9B 00000           CON(5)  =R>
1355 E3BA0 00000           CON(5)  =ADD            DP+#NIBS to ALLOT
1356 E3BA5 00000           CON(5)  =S0             S0  start of DATA STK
1357 E3BAA 00000           CON(5)  =LIT                now subtract off
```



```
1358 E3BAF 00000           CON(5)  458              448 nibs.
1359                *                           200 nibs for data stack,
1360                *                           168 (84 bytes) for PAD
1361                *                           90 nibs for DP-PAD buffer
1362                *
1363 E3BB4 00000           CON(5)  =MINUS           -
1364 E3BB9 00000           CON(5)  =GT              >  see if DP > S0-458
1365 E3BBE 78C3E           CON(5)  =ABRT"X          if so give error message
1366 E3BC3 80              CON(2)  =eDFUL           ABORT"dictionary full"
1367 E3BC5 00000           CON(5)   =PSTOR          +!
1368 E3BCA 00000           CON(5)   =SEMI           ;
```

```
1369                    EJECT
1370                ***********************************************
1371                *
1372                * ?COMP  ( --- ) Issue an error message if not
1373                *        compiling.
1374                *
1375                ***********************************************
1376
1377 E3BCF D5B3             CON(5)  =!ALOTB
          E
1378 E3BD4 58     =!?COMP   CON(2)  #85             ?COMP
1379 E3BD6 F334             NIBASC  \?COM\
          F4D4
1380 E3BDE 0b             CON(2)  \P\+#80
1381
1382 E3BE0 0000   =?COMP   CON(5)  =DOCOL          :
          0
1383 E3BE5 C183           CON(5)   =STATE          STATE
          E
1384 E3BEA 0000           CON(5)   =AT             @
          0
1385 E3BEF 0000           CON(5)   =ZEQ            0=
          0
1386 E3BF4 22E3           CON(5)   =ABORTW         ABORT"_
          E
1387 E3BF9 90             CON(2)  =eCOMP
1388               *      NIBASC  \ compil\
1389               *      NIBASC  \e only\
1390 E3BFB 0000           CON(5)   =SEMI           ;
          0
```



```
1391                    EJECT
1392                ***************************************************************
1393                *
1394                * -FIND  ( --- cfa  b tf) if found
1395                *        ( --- ff       ) if not found
1396                *
1397                * Accepts the next word, delimited by blanks, in the
1398                * input stream to HERE and searches the CONTEXT and then
1399                * the FORTH vocabularies for a matching entry.  NOTE:
1400                * in this rendition of -FIND we return the CFA and NOT the
1401                * PFA
1402                *
1403                * MADE HEADERLESS 10/10/83
1404                ***************************************************************
1405
1406
1407 E3C00 0000 =-FIND  CON(5)  =DOCOL          :
          0
1408 E3C05 0000           CON(5)   =BL             BL
          0
1409 E3C0A 0000           CON(5)   =WORD           WORD
          0
1410 E3C0F 0000           CON(5)   =CONTX          CONTEXT
          0
1411 E3C14 0000           CON(5)   =AT             @
          0
1412 E3C19 0000           CON(5)   =AT             @
          0
1413 E3C1E EA43           CON(5)   =FIND>          <FIND>
          E
1414 E3C23 0000           CON(5)   =SEMI           ;
          0
```

```
1415                    EJECT
1416
1417            ****************************************
1418            *
1419            * Utility routines for FORTH ABORT rtnes
1420            *
1421            ****************************************
1422
1423 E3C28 0000 =CLEANX CON(5)   =DOCOL
               0
1424 E3C2D 0000         CON(5)   =SP!           reset data stack
               0
1425 E3C32 E9CF         CON(5)  (=FRTHHD)+12    reset FORTH as context
               2
1426 E3C37 0000         CON(5)   =DEFIN         and current vocabulary
               0
1427 E3C3C 0000         CON(5)   =CLALL         close all open files
               0
1428 E3C41 E9E3         CON(5)   =CLEAN         clean up last entry if smudged
               E
1429 E3C46 7B23         CON(5)   =FLUSH         flush disc buffers
               E
1430 E3C4B 0000         CON(5)   =SEMI
               0
1431
1432 E3C50 0000 =ERRTST CON(5)   =DOCOL       This rtne tests for FORTH ON ERRORs
               0
1433 E3C55 BF73         CON(5)   =ONERR        If contents of ONERR <> 0 then
               E
1434 E3C5A 0000         CON(5)   =AT           execute the contents of ONERR as
               0
1435 E3C5F 0000         CON(5)   =?DUP         the user's ON ERROR rtne.  Just
               0
1436 E3C64 0000         CON(5)   =ZBRNH        in case they don't get it right
               0
1437 E3C69 F000         CON(5)   (ET0)-*       and return to here, we'll QUIT.
               0
1438 E3C6E 0000         CON(5)   =EXEC
               0
1439 E3C73 0000         CON(5)   =QUIT
               0
1440 E3C78 0000 ET0     CON(5)   =SEMI
               0
1441
1442
1443 E3C7D 82C3 AB04    CON(5)   =CLEANX
               E
1444 E3C82 0000         CON(5)   =QUIT
               0
1445
1446            ****************************************
1447            *
1448            * ABRT"X  This rtne is a FORTH abort for
1449            *         issuing error messages without
1450            *         appending anything to them (i.e.,
```



```
1451            *         FTH ERR: no ending "  )
1452            *
1453            ****************************************
1454
1455 E3C87 0000 =ABRT"X CON(5)   =DOCOL
               0
1456 E3C8C 0000         CON(5)   =ZBRNH     IF
               0
1457 E3C91 C300         CON(5)   (ABXX)-*
               0
1458 E3C96 0000         CON(5)  =R@         POINT TO ERR#
               0
1459 E3C9B 0000         CON(5)  =CAT        GET IT
               0
1460 E3CA0 E0D3         CON(5)  =SETERN     SET SYSTEM LOCATION ERR# IN CASE
               E
1461            *                           WE JUMP OUT TO ON-ERROR NEXT
1462 E3CA5 05C3         CON(5)   =ERRTST        TEST FOR (& do) ON ERROR
               E
1463            *                           continue if noi ON-ERROR
1464 E3CAA 0000         CON(5)   =ZERO          LEAVE FLAG FOR EMSG
               0
1465            *                           saying no word to insert (ABORTW)
1466 E3CAF 0000 AB01    CON(5)   =R@
               0
1467 E3CB4 0000         CON(5)   =CAT           GET ERROR NUMBER (again...we did
               0
1468            *                           save it so ON-ERROR routine wouldn't
1469            *                           have yet another thing on stack
1470 E3CB9 A6D3         CON(5)   =EMSG      Actually generate the message (and po
               E
1471 E3CBE 1EC3         CON(5)   =BASIC?
               E
1472            * If from FORTHX, BASIC? will go BYE.
1473 E3CC3 0000         CON(5)   =BRNCH
               0
1474 E3CC8 5BFF         CON(5)   (AB04)-*
               F
1475            *                           ELSE just move I past compiled msg#
1476 E3CCD 0000 ABXX    CON(5)   =R>
               0
1477 E3CD2 0000         CON(5)   =TWO+          SKIP THE MSG#
               0
1478 E3CD7 0000         CON(5)   =>R
               0
1479 E3CDC 0000         CON(5)   =SEMI
               0
1480
1481            * IF FROM BASIC, GO BYE-BYE
1482 E3CE1 0000 =BASIC? CON(5)   =DOCOL
               0
1483 E3CE6 1B54         CON(5)   =BASIC
               E
1484 E3CEB 0000         CON(5)   =AT
               0
```

```
1485 E3CF0 0000          CON(5)   =ZBRNH          IF from BASIC
           0
1486 E3CF5 4100          CON(5)   (BASEND)-*
           0
1487 E3CFA 0000          CON(5)   =SP!            reset data stack
           0
1488 E3CFF A3D3          CON(5)   =BTFERR         set flag (oERR?) for BYE;clear (oBASI
           E
1489             *                                put return address (in oBASIC) on sta
1490             *                                BSCRTN
1491 E3D04 0000          CON(5)   =BSCRTN         jump to addr on stack
           0
1492 E3D09 0000 BASEND   CON(5)   =SEMI
           0
1493
1494
1495             * SET ERRN in mainframe
1496 E3D0E 31D3 =SETERN  CON(5)   =$STERN
           E
1497 E3D13 3400 =$STERN  LC(5)    =ERR#          SYSTEM LOCATION
           000
1498 E3D1A 136           CD0EX
1499 E3D1D 108           R0=C                     SAVE D0
1500 E3D20 14F           C=DAT1   B               GET OUR ERROR# ON STACK
1501 E3D23 174           D1=D1+   5               DROP IT
1502 E3D26 22            P=       2
1503 E3D28 31F2          LC(2)    =ID             C HAS ID + ERR#
1504 E3D2C 20            P=       0
1505 E3D2E 15C3          DAT0=C   4               STORE FOR SYSTEM
1506 E3D32 118           C=R0                     GET BACK D0
1507 E3D35 134           D0=C
1508 E3D38 03            RTNCC
1509
1510
1511 E3D3A F3D3 =BTFERR  CON(5)   =$BTFER
           E
1512 E3D3F 132  =$BTFER  AD0EX
1513 E3D42 1BBB          D0=(5)   =oERR?          BYE SETS NOCONT IF ERR? SET
           BF2
1514 E3D49 D2            C=0      A
1515 E3D4B E6            C=C+1    A
1516 E3D4D 144           DAT0=C   A                   SET oERR? FLAG
1517 E3D50 1B2A          D0=(5)   =oBASIC             GET RTN ADDR
           BF2
1518 E3D57 146           C=DAT0   A
1519 E3D5A 1C4           D1=D1-   5
1520 E3D5D 145           DAT1=C   A                   PUT RTN ON STACK FOR BSCRTN
1521 E3D60 D2            C=0      A
1522 E3D62 144           DAT0=C   A                   CLEAR BASIC FLAG
1523 E3D65 132           AD0EX                        RESTORE D0
1524 E3D68 03            RTNCC
1525             *
1526             * Error number is on the stack. Generate its error message.
1527             * Stack also has flag to indicate simple ABORT or ABORTW (with
1528             * word at HERE embedded in message).
```



```
1529 E3D6A F6D3 =EMSG    CON(5)   =$EMSG
           E
1530 E3D6F 8E00 =$EMSG   GOSUBL   =SAVEFP
           00
1531 E3D75 AF2           C=0      W
1532 E3D78 14F           C=DAT1   B               GET ERROR #
1533 E3D7B AE7           D=C      B               SAVE IT
1534 E3D7E 20            P=       0
1535 E3D80 174           D1=D1+   5               POINT TO WHERE-FLAG
1536 E3D83 143           A=DAT1   A               GET IT
1537 E3D86 8A8           ?A=0     A               SIMPLE ABORT?
1538 E3D89 03            GOYES    EMS01
1539 E3D8B 174           D1=D1+   5               GET "HERE"
1540 E3D8E 143           A=DAT1   A                TO GET WORD IN DICTIONARY
1541 E3D91 131           D1=A
1542 E3D94 E4            A=A+1    A
1543 E3D96 E4            A=A+1    A               POINT PAST COUNT
1544 E3D98 D6            C=A      A               SAVE TEMPORARILY
1545 E3D9A AF0           A=0      W
1546 E3D9D 14B           A=DAT1   B               GET MSG LEN
1547 E3DA0 A6C           A=A-1    B               CHAR-1 FOR MFWRN
1548 E3DA3 814           ASRC
1549 E3DA6 814           ASRC                     PUT IN A[15-14]
1550 E3DA9 DA            A=C      A               GET BACK MSG ADDR
1551 E3DAB 102           R2=A                     R2=INSERT ADDR FOR MFWRN
1552 E3DAE AF2           C=0      W
1553 E3DB1 AEB           C=D      B               GET BACK MSG #
1554 E3DB4 2D            P=       13
1555 E3DB6 B06           C=C+1    P               C[13]=1
1556 E3DB9 22   EMS01    P=       2
1557 E3DBB 31F2          LC(2)    =ID             LEX ID
1558 E3DBF 108           R0=C
1559             * SET ERR? TO TELL ERROR-POLL-HANDLER TO RETURN
1560 E3DC2 1BBB          D0=(5)   =oERR?
           BF2
1561 E3DC9 144           DAT0=C   A
1562 E3DCC 8F00          GOSBVL   =POLL
           000
1563 E3DD3 00            CON(2)   =pERROR
1564 E3DD5 80FF          CPEX     15
1565 E3DD9 2C            P=       12
1566 E3DDB 32F0          LCHEX    00F
           0
1567 E3DE0 480           GOC      EMS02           ERR IN POLL
1568 E3DE3 118           C=R0
1569 E3DE6 30F           LCHEX    F
1570 E3DE9 8E00 EMS02    GOSUBL   =GTS13          SET S13 IF PROGRAM IS RUNNING
           00
1571             *                                SO ON-ERROR CAN HAPPEN
1572             *                                ONLY A&D0 CHANGED (&S13)
1573 E3DEF AF7           D=C      W               SAVE MSG INFO
1574 E3DF2 8E00          GOSUBL   =GETFP          GET D1,D0,B
           00
1575 E3DF8 8E00          GOSUBL   =$SP!           RESET D1 TO EMPTY STACK
           00
```

```
1576           *                          (WOULD BE DONE LATER, UNLESS
1577           *                           ON-ERROR TAKES OVER NOW)
1578 E3DFE 8E00        GOSUBL  =SAVEFP     NEW TOS IN DSTKAD
          00
1579 E3E04 AFB         C=D     W           RESTORE MSG INFO
1580 E3E07 D0          A=0     A
1581 E3E09 1BBB        D0=(5)  =oERR?
          BF2
1582 E3E10 140         DAT0=A  A             CLEAR ERR?
1583           *       GOSBVL  =MFERsp       SHOW MSG
1584 E3E13 8F          NIBHEX  8F
1585 E3E15 D049        CON(5)  MFERsp
          0
1586 E3E1A 8E00        GOSUBL  =GETFP
          00
1587 E3E20 03          RTNCC
1588
1589           ******************************************
1590           *
1591           * ABORTW  is a FORTH internal abort for
1592           *         use in cases where we wish to
1593           * preface an error message with the current
1594           * word (i.e.,  FTH ERR: XYZ not recognized)
1595           *
1596           ******************************************
1597
1598 E3E22 0000 =ABORTW CON(5)   =DOCOL       if zero flag skip this
          0
1599 E3E27 0000        CON(5)   =ZBRNH       abort routine
          0
1600 E3E2C 1AEF        CON(5)   (ABXX)-*
          F
1601 E3E31 0000        CON(5)  =R@           POINT TO ERROR#
          0
1602 E3E36 0000        CON(5)  =CAT          GET IT
          0
1603 E3E3B E0D3        CON(5)  =SETERN       SET SYSTEM VARIABLE
          E
1604 E3E40 05C3        CON(5)   =ERRTST        TEST FOR (& DO) ON ERROR
          E
1605 E3E45 0000        CON(5)   =HERE          ADDR OF COUNTED STRING FOR DISPLAY
          0
1606           *                                used by EMSG
1607 E3E4A 0000        CON(5)   =ONE           SIGNAL TO EMSG that string to inse
          0
1608 E3E4F 0000        CON(5)   =BRNCH         REPLICATE CODE IN ABRT"X
          0
1609 E3E54 B5EF        CON(5)   (AB01)-*
          F
1610
1611           * get addr of variable, just like with DOUSE, simplified for
1612           * call from within primitive
1613 E3E59 E5E3 =ACTIV  CON(5)   =$ACTIV
          E
1614 E3E5E 20    =$ACTIV P=      0
```



```
1615 E3E60 34C0        LC(5)    =ACTIVE
          BF2
1616 E3E67 1C4         D1=D1-   5
1617 E3E6A 145         DAT1=C   A
1618 E3E6D 03          RTNCC
1619
1620           ******************************************************
1621           *
1622           * ABORT Clears the data and return stacks, setting
1623           *       execution mode. Closes all open files, flushes
1624           *       disc buffers, but does not clear SCRFIB or LINE#
1625           *       so that they may be used for error tracking
1626           *       purposes. Returns control to
1627           *       the terminal.
1628           *
1629           *       If ONERR is non-zero will use contents of ONERR
1630           * as CFA of ON ERROR routine, will not reset data or rtn
1631           * stacks, user must take care of that.
1632           *
1633           *       Although not standard FORTH ABORT also checks
1634           * to see if the last word in the dictionary is still
1635           * smudged.  If it is smudged it is a bad definition and
1636           * so ABORT by way of CLEAN will clean this stuff out of
1637           * the dictionary by resetting the DP.
1638           *
1639           ******************************************************
1640 E3E6F 4DB3        CON(5)   =!?COMP
          E
1641 E3E74 58    =!ABORT CON(2)   #85
1642 E3E76 1424        NIBASC   \ABOR\
          F425
1643 E3E7E 4D          CON(2)   \T\+#80
1644
1645 E3E80 0000 =ABORT  CON(5)  =DOCOL          :
          0
1646 E3E85 05C3        CON(5)  =ERRTST         TEST FOR (& DO) ON ERROR RTNE
          E
1647 E3E8A 82C3        CON(5)  =CLEANX
          E
1648 E3E8F 1EC3 ABR00  CON(5)   =BASIC?         IF COMING FROM BASIC
          E
1649           *                                NEVER RETURN
1650 E3E94 0000 ABRT1  CON(5)   =QUIT
          0
1651 E3E99 0000        CON(5)   =SEMI           ;
          0
1652
1653
1654
1655 E3E9E 0000 =CLEAN  CON(5)   =DOCOL          :
          0
1656 E3EA3 0000        CON(5)   =LATE           LATEST  get NFA of
          0
1657 E3EA8 0000        CON(5)   =DUP            DUP     latest word, copy it
          0
```

```
1658 E3EAD 0000          CON(5)   =CAT            C@      get 1st byte of name fiel
           0
1659 E3EB2 0000          CON(5)   =LIT                    and in hex #20 to see
           0
1660 E3EB7 0200          CON(5)   #20             #20     if word is smudged.
           0
1661 E3EBC 0000          CON(5)   =AND            AND     If so
           0
1662 E3EC1 0000          CON(5)   =ZBRNH          IF      its garbage so lets recla
           0
1663 E3EC6 2300          CON(5)   (ABRT0)-*               this space for the user
           0
1664 E3ECB 0000          CON(5)   =FIVE-          5-      back up NFA addr to point
           0
1665 E3ED0 0000          CON(5)   =DUP            DUP     save copy for CURRENT bac
           0
1666 E3ED5 0000          CON(5)   =DP             DP      link field and store this
           0
1667 E3EDA 0000          CON(5)   =STORE          !       as the new dictionary poi
           0
1668 E3EDF 0000          CON(5)   =AT             @       now get last link
           0
1669 E3EE4 0000          CON(5)   =CURR           CURRENT get addr of current voca
           0
1670 E3EE9 0000          CON(5)   =AT             @        pointer
           0
1671 E3EEE 0000          CON(5)   =STORE          !        reset current vocab to 1
           0
1672 E3EF3 0000          CON(5)   =ONE            1       phoney value which is dro
           0
1673           *                                          it just saves time and co
1674 E3EF8 0000 ABRT0    CON(5)   =DROP           DROP    drop copy of NFA
           0
1675 E3EFD 0000          CON(5)   =SEMI
           0
1676
```



```
1677                     EJECT
1678           ********************************************************
1679           *
1680           * WHERE   Display the last char string parsed by the text
1681           *
1682           ********************************************************
1683
1684 E3F02 0000 =WHERE  CON(5)   =DOCOL          :
           0
1685 E3F07 0000          CON(5)   =CR             CR
           0
1686 E3F0C 0000          CON(5)   =HERE           HERE
           0
1687 E3F11 0000          CON(5)   =DUP            here here
           0
1688 E3F16 0000          CON(5)   =TWO+           here here+2
           0
1689 E3F1B 0000          CON(5)   =SWAP           here+2 here
           0
1690 E3F20 0000          CON(5)   =CAT            here+2 char-count
           0
1691 E3F25 0000          CON(5)   =ZERO           here+2 count 0
           0
1692 E3F2A 0000          CON(5)   =XDO            lenght-of-string 0 DO
           0
1693 E3F2F 0000 WH01     CON(5)   =DUP            char-ptr char-ptr
           0
1694 E3F34 0000          CON(5)   =CAT            char-ptr char
           0
1695 E3F39 0000          CON(5)   =EMIT           EMIT
           0
1696 E3F3E 0000          CON(5)   =TWO+           2+      MOVE TO NEXT CHAR
           0
1697 E3F43 1FF3          CON(5)   =XLOOP          LOOP
           E
1698 E3F48 7EFF          CON(5)   (WH01)-*
           F
1699 E3F4D 0000          CON(5)   =DROP           drop char-ptr
           0
1700 E3F52 D6F3          CON(5)   =SPACE          SPACE
           E
1701 E3F57 0000          CON(5)   =SEMI           ;
           0
```

```
1702                    EJECT
1703                ************************************************************
1704                *
1705                * SPACE  Transmit an ASCII blank to the current output device.
1706                *
1707                ************************************************************
1708 E3F5C 47E3            CON(5)  =!ABORT
          E
1709 E3F61 58      =!SPACE CON(2)  #85             SPACE
1710 E3F63 3505            NIBASC  \SPAC\
          1434
1711 E3F6B 5C              CON(2)  \E\+#80
1712
1713 E3F6D 0000    =SPACE  CON(5)  =DOCOL          :
          0
1714 E3F72 0000            CON(5)   =BL             BL
          0
1715 E3F77 0000            CON(5)   =EMIT           EMIT
          0
1716 E3F7C 0000            CON(5)   =SEMI           ;
          0
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER


```
1717                    EJECT
1718                ************************************************************
1719                *
1720                * <+LP>  Runtime code for +LOOP    (n --- )
1721                *
1722                *  The loop is supposed to continue until I crosses
1723                *  the boundary between L-1 and L.  However in +LOOP
1724                *  it can cross the boundary in either direction.
1725                *  The cases break down, however, in the following fashion:
1726                *
1727                *  On entry to <+LP> code:
1728                *
1729                *  INDEX >= LIMIT    If INDEX < LIMIT after increment then
1730                *                    stop loop
1731                *
1732                *  INDEX < LIMIT     If INDEX >= LIMIT after increment then
1733                *                    stop loop
1734                *
1735                ************************************************************
1736
1737 E3F81 68F3 =<+LP>  CON(5) =$<+LP>
          E
1738
1739 E3F86 143  =$<+LP> A=DAT1  A             READ n TO A(A)
1740 E3F89 101          R1=A                  SAVE INCREMENT IN R1
1741 E3F8C 174          D1=D1+  5             RTN NOTHING ON DATA STK
1742 E3F8F D9           C=B     A
1743 E3F91 137          CD1EX                 D1=RTN STK
1744 E3F94 D7           D=C     A             SAVE DATA STK PTR IN D(A)
1745 E3F96 147          C=DAT1  A             C(A)=LOOP INDEX
1746 E3F99 174          D1=D1+  5
1747 E3F9C 143          A=DAT1  A             A(A)=LOOP LIMIT
1748 E3F9F 1C4          D1=D1-  5             PT TO INDEX FOR UPDATE LATER
1749 E3FA2 7A20         GOSUB   LT            SEE IF INDEX < LIMIT
1750 E3FA6 5D0          GONC    I>=L          NO, I >= L
1751 E3FA9 7210         GOSUB   BEFLT         RESTORE I,L; ADD INCR AND TEST
1752 E3FAD 446          GOC     CONT          I STILL < L SO CONTINUE LOOP
1753 E3FB0 6970         GOTO    STOP          ELSE I>=L SO STOP LOOP
1754
1755 E3FB4 7700 I>=L    GOSUB   BEFLT         RESTORE I,L; ADD INCR AND TEST
1756 E3FB8 595          GONC    CONT          I STILL >=L SO CONTINUE LOOP
1757 E3FBB 6E60         GOTO    STOP          ELSE I < L SO STOP LOOP
1758
1759 E3FBF 81E  BEFLT   CSRB                  SHIFT INDEX INTO C(A)
1760 E3FC2 81C          ASRB                  SHIFT LIMIT INTO A(A)
1761 E3FC5 121          AR1EX                 R1=LIMIT, A(A)=INCREMENT
1762 E3FC8 C2           C=C+A   A             NEW INDEX VALUE
1763 E3FCA 145          DAT1=C  A             STORE NEW INDEX VALUE TO RTN STK
1764 E3FCD 111          A=R1                  A(A)=LIMIT
1765
1766 E3FD0 25   LT      P=      5
1767 E3FD2 A80          A=0     P             CLEAR FOR NEGATIVE TEST
1768 E3FD5 A82          C=0     P
1769 E3FD8 A14          A=A+A   WP            IF LIMIT NEG THEN A(5)=1
1770 E3FDB A16          C=C+C   WP            IF INDEX NEG THEN A(5)=1
```

```
1771 E3FDE 902          ?C=A    P           ARE SIGNS THE SAME?
1772 E3FE1 90           GOYES   SAMSGN      IF SO GO HERE
1773 E3FE3 992          ?A<C    WP
1774 E3FE6 00           RTNYES              DIFFERENT SIGNS, SWITCH SENSE OF TE
1775 E3FE8 03           RTNCC
1776
1777 E3FEA 996  SAMSGN  ?A>C    WP          IS C<A?
1778 E3FED 00           RTNYES              IF SO RTN WITH CARRY SET
1779 E3FEF 03           RTNCC               ELSE RTN WITH CARRY CLEAR
1780
1781            *********************************
1782            *
1783            * XLOOP Runtime code for LOOP
1784            *
1785            *********************************
1786
1787 E3FF1 6FF3 =XLOOP  CON(5) =$XLOOP
         E
1788
1789 E3FF6 D9   =$XLOOP C=B     A
1790 E3FF8 137          CD1EX                  RTN STK TO D1
1791 E3FFB D7           D=C     A              SAVE DATA STK IN D
1792 E3FFD 143          A=DAT1  A              A=INDEX
1793 E4000 E4           A=A+1   A              INCREMENT INDEX
1794
1795 E4002 141  XLP00   DAT1=A  A              PUSH NEW INDEX
1796 E4005 174          D1=D1+  5              PT TO LIMIT
1797 E4008 147          C=DAT1  A              LIMIT
1798 E400B E2           C=C-A   A              LIMIT-INDEX
1799 E400D 8AA          ?C=0    A              DONE?
1800 E4010 D1           GOYES   XLP0           YES
1801 E4012 142  CONT    A=DAT0  A              READ OFFSET
1802 E4015 136          CD0EX                  I TO C
1803 E4018 C2           C=C+A   A              NEW I
1804 E401A 134          D0=C                   RESTORE I
1805 E401D DB           C=D     A
1806 E401F 135          D1=C                   RESTORE DATA STK PTR
1807 E4022 8E00         GOSUBL  =DOATN         CHECK FOR ATTN KEY
         00
1808 E4028 03           RTNCC
1809
1810 E402A 174  STOP    D1=D1+  5
1811
1812 E402D DB   XLP0    C=D     A              RECOVER DATA STK PTR TO C(A)
1813 E402F 174          D1=D1+  5              DROP LIMIT FROM RTN STK
1814 E4032 137          CD1EX                  D1=DATA STK, C=RTN STK
1815 E4035 D5           B=C     A              RESTORE RTN STK
1816 E4037 164          D0=D0+  5              NEW I
1817 E403A 8E00         GOSUBL  =DOATN         CHECK FOR ATTN KEY
         00
1818 E4040 03           RTNCC                  DOATN REQUIRES GOSUB/RTNCC
```



```
1819            EJECT
1820            ************************************************************
1821            *
1822            * DEPTH ( --- n) Leave # of entries on stack, before DEPTH
1823            *       was executed.
1824            *
1825            ************************************************************
1826
1827 E4042 16F3          CON(5)  =!SPACE
         E
1828 E4047 58   =!DEPTH  CON(2)  #85            DEPTH
1829 E4049 4454          NIBASC  \DEPT\
         0545
1830 E4051 8C            CON(2)  \H\+#80
1831
1832 E4053 0000 =DEPTH   CON(5)  =DOCOL         :
         0
1833 E4058 0000          CON(5)   =S0           S0    stack bottom
         0
1834 E405D 0000          CON(5)   =SP@          SP@   current stack position
         0
1835 E4062 0000          CON(5)   =MINUS        -     get difference
         0
1836 E4067 0000          CON(5)   =LIT
         0
1837 E406C 5000          CON(5)   5             5     get # addresses
         0
1838 E4071 0000          CON(5)   =/            /
         0
1839 E4076 0000          CON(5)   =ONE-         1-    we were 1 off due to S0
         0
1840 E407B 0000          CON(5)   =SEMI         ;
         0
1841
1842
1843            ************************************************************
1844            *
1845            * */MOD (n1 n2 n3 --- n4 n5) Multiply n1 by n2, divide the
1846            *       result by n3, leave the remainder n4 and quotient n5.
1847            *
1848            ************************************************************
1849
1850 E4080 7404          CON(5)  =!DEPTH
         E
1851 E4085 58   =!*/MOD  CON(2)  #85            */MOD
1852 E4087 A2F2          NIBASC  \*/MO\
         D4F4
1853 E408F 4C            CON(2)  \D\+#80
1854
1855 E4091 0000 =*/MOD   CON(5)  =DOCOL         :
         0
1856 E4096 0000          CON(5)   =>R           >R
         0
1857 E409B 0000          CON(5)   =M*           M*
         0
```

```
1858 E40A0 0000          CON(5)   =R>          R>
           0
1859 E40A5 0000          CON(5)   =M/          M/
           0
1860 E40AA 0000          CON(5)   =SEMI        ;
           0
```



```
1861                     EJECT
1862                *****************************************************************
1863                *
1864                * M/MOD  (ud1 u2 --- u3 ud4) Unsigned mixed magnitude math
1865                *        operation. Leaves remainder u3 and double quotient ud4.
1866                *
1867                *****************************************************************
1868 E40AF 5804            CON(5)   =!*/MOD
           E
1869 E40B4 58    =!M/MOD   CON(2)   #85              M/MOD
1870 E40B6 D4F2            NIBASC   \M/MO\
           D4F4
1871 E40BE 4C              CON(2)   \D\+#80
1872
1873 E40C0 5C04 =M/MOD     CON(5)   $M/MOD
           E
1874 E40C5 8E00 =$M/MOD    GOSUBL    =$UDIV      WRITES REMAINDER & LSD OF QUOTIENT
           00
1875 E40CB 1C9             D1=D1-    10
1876 E40CE 15D9            DAT1=C    10          WRITE MSD OF QUOTIENT
1877 E40D2 174             D1=D1+    5
1878 E40D5 03              RTNCC
```

```
1879                    EJECT
1880                ************************************************
1881                *
1882                * <;CODE> (FOR ROM PROVIDED DEFINING WORDS)
1883                *         This word stores the runtime address of
1884                *         the appropriate prologue.
1885                *
1886                ************************************************
1887
1888 E40D7 0000 =ROM;C  CON(5)  =DOCOL          :
           0
1889 E40DC 0000          CON(5)   =R>           R>
           0
1890 E40E1 0000          CON(5)   =AT           @
           0
1891 E40E6 0000          CON(5)   =LATE         LATEST
           0
1892 E40EB 0000          CON(5)   =PFA          PFA
           0
1893 E40F0 0000          CON(5)   =LIT
           0
1894 E40F5 5000          CON(5)  5              5
           0
1895 E40FA 0000          CON(5)   =MINUS        -
           0
1896 E40FF 0000          CON(5)   =STORE        !
           0
1897 E4104 0000          CON(5)   =SEMI         ;
           0
```



```
1898                    EJECT
1899                ************************************************
1900                *
1901                * ENDOF  (FROM CASE  OF  ENDOF  ENDCASE)
1902                *
1903                ************************************************
1904
1905 E4109 4B04           CON(5)  =!M/MOD
           E
1906 E410E 5C     =!ENDOF CON(2)  #C5
1907 E4110 54E4           NIBASC  \ENDO\
           44F4
1908 E4118 6C             CON(2)  \F\+#80
1909
1910 E411A 0000 =ENDOF CON(5) =DOCOL        :
           0
1911 E411F 0EB3           CON(5)   =?COMP     ensure compile mode
           E
1912
1913                * start with ADDR 5 left by of
1914
1915 E4124 0000           CON(5)   =LIT
           0
1916 E4129 5000           CON(5)  5            from OF
           0
1917 E412E 0000           CON(5)   =?PAIR      error if no 5 on stack
           0
1918 E4133 0000           CON(5)   =COMPL      COMPILE
           0
1919 E4138 0000           CON(5)   =BRNCH      BRANCH
           0
1920 E413D 0000           CON(5)   =HERE       HERE
           0
1921                * ADDR HERE
1922 E4142 0000           CON(5)   =ZERO       0
           0
1923 E4147 0000           CON(5)   =COMMA      allot space for jump to be filled in
           0
1924 E414C 0000           CON(5)   =SWAP       HERE ptr-to-IF
           0
1925                * HERE ADDR
1926 E4151 0000           CON(5)   =TWO        TWO
           0
1927                *      CON(5)   =[CMP]      [COMPILE]
1928 E4156 0000           CON(5)   =THEN       fill in HERE at ptr left by IF to its
           0
1929                * HERE
1930 E415B 0000           CON(5)   =LIT
           0
1931 E4160 7000           CON(5)   7           leave 7 for syntax checking
           0
1932 E4165 0000           CON(5)   =SEMI       ;
           0
```

```
1933                    EJECT
1934                ****************************************************************
1935                *
1936                * BLOCK  (n --- addr)  This routine leaves the address of the
1937                *        first byte in line# n from the file identified by the
1938                *        fib# contained in SCRFIB. If the line is not already in
1939                *        a buffer it is transferred from the file (which may be
1940                *        in RAM or on mass storage).
1941                *
1942                ****************************************************************
1943
1944 E416A E014         CON(5)  =!ENDOF
           E
1945 E416F 58    =!BLOCK CON(2) #85            BLOCK
1946 E4171 24C4         NIBASC \BLOC\
           F434
1947 E4179 BC           CON(2) \K\+#80
1948
1949 E417B 0000 =BLOCK  CON(5)  =DOCOL          :
           0
1950 E4180 0000        CON(5)   =>R            >R      save line# on rtn stk
           0
1951 E4185 0000        CON(5)   =PREV          PREV    get addr of previously
           0
1952 E418A 0000        CON(5)   =AT            @       used buffer
           0
1953 E418F 0000        CON(5)   =DUP           DUP     duplicate it
           0
1954 E4194 0000        CON(5)   =CAT           C@      get its fib#
           0
1955 E4199 0000        CON(5)   =SCFIB         SCRFIB  get the currently
           0
1956 E419E 0000        CON(5)   =AT            @       active fib#
           0
1957 E41A3 0000        CON(5)   =MINUS         -       if they are not
           0
1958 E41A8 0000        CON(5)   =ZBRNH         IF      the same then
           0
1959 E41AD 6900        CON(5)  (BLK1)-*
           0
1960 E41B2 0000 BLK00  CON(5)   =PBUF          BEGIN +BUF  get addr of next
           0
1961 E41B7 0000        CON(5)   =ZEQ           0=      buffer, if its the
           0
1962 E41BC 0000        CON(5)   =ZBRNH         IF      same as the PREV buffer
           0
1963 E41C1 B400        CON(5)  (BLK10)-*
           0
1964 E41C6 0000        CON(5)   =DROP          DROP    then drop PREV buff addr
           0
1965 E41CB 0000        CON(5)   =R@            R@      get line # from rtn stk
           0
1966 E41D0 0000        CON(5)   =SCFIB         SCRFIB  get active fib#
           0
1967 E41D5 0000        CON(5)   =AT            @
```



```
           0
1968 E41DA 0000        CON(5)   =BUFFR         BUFFER  and get a buffer for this
           0
1969 E41DF 0000        CON(5)   =DUP           DUP     duplicate buffer DATA add
           0
1970 E41E4 0000        CON(5)   =R@            R@      get line # from rtn stk
           0
1971 E41E9 0000        CON(5)   =SCFIB         SCRFIB  get active fib#
           0
1972 E41EE 0000        CON(5)   =AT            @
           0
1973 E41F3 0000        CON(5)   =ONE           1       set read flag
           0
1974 E41F8 0000        CON(5)   =R/W           R/W     read this line#
           0
1975 E41FD 0000        CON(5)   =LIT
           0
1976 E4202 B000        CON(5)   11             11
           0
1977 E4207 0000        CON(5)   =MINUS         -       get buffer start addr
           0
1978 E420C 0000 BLK10  CON(5)   =DUP           DUP     duplicate it
           0
1979 E4211 0000        CON(5)   =CAT           C@      fetch its fib#
           0
1980 E4216 0000        CON(5)   =SCFIB         SCRFIB  get current fib#
           0
1981 E421B 0000        CON(5)   =AT            @
           0
1982 E4220 0000        CON(5)   =MINUS         -       compare and if
           0
1983 E4225 0000        CON(5)   =ZEQ           0=      not the same go again
           0
1984 E422A 0000        CON(5)   =ZBRNH         UNTIL
           0
1985 E422F 38FF        CON(5)  (BLK00)-*
           F
1986 E4234 0000        CON(5)   =DUP           DUP     duplicate buffer addr
           0
1987 E4239 0000        CON(5)   =PREV          PREV    store it in PREV
           0
1988 E423E 0000        CON(5)   =STORE         !
           0
1989 E4243 0000 BLK1   CON(5)   =DUP           THEN DUP   duplicate buffer addr
           0
1990 E4248 0000        CON(5)   =TWO+          2+      get line# addr in buffer
           0
1991 E424D 0000        CON(5)   =AT            @       get line#
           0
1992 E4252 0000        CON(5)   =R@            R@      get desired line# from rt
           0
1993 E4257 0000        CON(5)   =MINUS         -       compare
           0
1994 E425C 0000        CON(5)   =ZBRNH         IF      if not the same
           0
```

```
1995 E4261 B400        CON(5)   (BLK2)-*              then dup buffer addr
           0
1996            *
1997            * In other FORTHs where a block is the whole file, BUFFER updates
1998            * the block number in the buffer.  Here where we have an unknown
1999            * number of lines per file and a block is a line we must update
2000            * the buffer in BLOCK to show the current line in the buffer.
2001            *
2002 E4266 0000        CON(5)   =DUP
           0
2003 E426B 0000        CON(5)   =TWO+                get line# addr in buffer
           0
2004 E4270 0000        CON(5)   =R@                  get desired line#
           0
2005 E4275 0000        CON(5)   =SWAP
           0
2006 E427A 0000        CON(5)   =STORE               set buffer for desired li
           0
2007 E427F 0000        CON(5)   =DUP          DUP
           0
2008 E4284 0000        CON(5)   =LIT                  make it buff data addr
           0
2009 E4289 B000        CON(5)   11            11
           0
2010 E428E 0000        CON(5)   =ADD          +
           0
2011 E4293 0000        CON(5)   =R@           R@      get desired line #
           0
2012 E4298 0000        CON(5)   =SCFIB        SCRBIF  get active fib#
           0
2013 E429D 0000        CON(5)   =AT           @
           0
2014 E42A2 0000        CON(5)   =ONE          1       set for read
           0
2015 E42A7 0000        CON(5)   =R/W          R/W THEN   read that line
           0
2016 E42AC 0000 BLK2   CON(5)   =R>           R>   get line# off rtn stk
           0
2017 E42B1 F783        CON(5)   =LINE#        this is necessary in case we had
           E
2018 E42B6 0000        CON(5)   =STORE        line we were looking for, already
           0
2019            *                             a buffer and R/W (which usually s
2020            *                             LINE#) was not called.
2021
2022 E42BB 0000        CON(5)   =LIT          buffer start addr still on data s
           0
2023 E42C0 B000        CON(5)   11            11    make it buffer data addr
           0
2024 E42C5 0000        CON(5)   =ADD          +
           0
2025 E42CA 0000        CON(5)   =SEMI         ;     return!
           0
```



```
2026            EJECT
2027            ********************************************************************
2028            *
2029            * LOADF   (addr  length ---- )  This routine assumes that
2030            *      the stack contains a character count and an address of
2031            *      a string describing the file to be loaded.
2032            *
2033            ********************************************************************
2034 E42CF F614        CON(5)  =!BLOCK
           E
2035 E42D4 58   =!LOADF CON(2) #85             LOADF
2036 E42D6 C4F4        NIBASC \LOAD\
           1444
2037 E42DE 6C          CON(2) \F\+#80
2038
2039
2040 E42E0 0000 =LOADF CON(5)   =DOCOL         :
           0
2041 E42E5 0000        CON(5)   =ABS           ABS VALUE OF LENGTH
           0
2042 E42EA 0000        CON(5)   =LIT
           0
2043 E42EF 2300        CON(5)   50
           0
2044 E42F4 0000        CON(5)   =MIN           allow max len name of 50 chars
           0
2045 E42F9 0000        CON(5)    =BLK          BLK    save current block#
           0
2046 E42FE 0000        CON(5)    =AT           @      on rtn stk
           0
2047 E4303 0000        CON(5)    =>R           >R
           0
2048 E4308 0000        CON(5)    =IN           >IN    save current pointer into
           0
2049 E430D 0000        CON(5)    =AT           @      input stream on rtn stk
           0
2050 E4312 0000        CON(5)    =>R           >R
           0
2051 E4317 0000        CON(5)    =SCFIB        SCRFIB   save current active fib#
           0
2052 E431C 0000        CON(5)    =AT           @        on rtn stk
           0
2053 E4321 0000        CON(5)    =>R           >R
           0
2054 E4326 F783        CON(5)    =LINE#        also save LINE# on rtn stk
           E
2055 E432B 0000        CON(5)    =AT
           0
2056 E4330 0000        CON(5)    =>R
           0
2057 E4335 0000        CON(5)    =ZERO         0       set pointer into input
           0
2058 E433A 0000        CON(5)    =IN           >IN     stream to 0
           0
2059 E433F 0000        CON(5)    =STORE        !
```

```
          0
2060 E4344 0000        CON(5)   =ONE          1       set desired line# to 1
          0
2061 E4349 0000        CON(5)   =BLK          BLK
          0
2062 E434E 0000        CON(5)   =STORE        !
          0
2063 E4353 0000        CON(5)   =ZERO         0
          0
2064 E4358 F783        CON(5)   =LINE#        LINE#   set line# associated with
          E
2065 E435D 0000        CON(5)   =STORE        !       active fib# to 0
          0
2066
2067 E4362 0144        CON(5)   =OPNF         OPENF    file, (it sets SCRFIB)
          E
2068 E4367 0000        CON(5)   =ZEQ          0=       if open was
          0
2069 E436C 0000        CON(5)   =ZBRNH        IF       unsuccessful then
          0
2070 E4371 1100        CON(5)   (LOD0)-*
          0
2071 E4376 D713        CON(5)   =ABORTN       move name to HERE supply 1
          E
2072 E437B 22E3        CON(5)   =ABORTW       for abort
          E
2073 E4380 73          CON(2)   =eCNLD        <name>  :cannot load
2074
2075 E4382 0000 LOD0   CON(5)   =EOF          don`t INTERPRET if empty file
          0
2076 E4387 0000        CON(5)   =ZEQ          reverse sense of EOF flag for 0BR
          0
2077 E438C 0000        CON(5)   =ZBRNH
          0
2078 E4391 A000        CON(5)   (LOD1)-*
          0
2079 E4396 0000        CON(5)   =INTRP        INTERPRET     interpret line
          0
2080 E439B 0000 LOD1   CON(5)   =SCFIB               let`s close this file
          0
2081 E43A0 0000        CON(5)   =AT
          0
2082 E43A5 0000        CON(5)   =CLOSF
          0
2083 E43AA 0000        CON(5)   =ZERO         and let's set this buffer to 0 fi
          0
2084 E43AF 0000        CON(5)   =PREV
          0
2085 E43B4 0000        CON(5)   =AT
          0
2086 E43B9 0000        CON(5)   =STORE
          0
2087 E43BE 0000        CON(5)   =R>           R>       now we'll restore old
          0
2088 E43C3 F783        CON(5)   =LINE#        LINE#   line # for old loadf
```



```
          E
2089 E43C8 0000        CON(5)   =STORE        !
          0
2090 E43CD 0000        CON(5)   =R>           R>      we're done, restore old fi
          0
2091 E43D2 0000        CON(5)   =SCFIB        SCRFIB
          0
2092 E43D7 0000        CON(5)   =STORE        !
          0
2093 E43DC 0000        CON(5)   =R>           R>    restore old pointer into
          0
2094 E43E1 0000        CON(5)   =IN           >IN   input stream
          0
2095 E43E6 0000        CON(5)   =STORE        !
          0
2096 E43EB 0000        CON(5)   =R>           R>     restore old desired line
          0
2097 E43F0 0000        CON(5)   =BLK          BLK    #
          0
2098 E43F5 0000        CON(5)   =STORE        !
          0
2099 E43FA 0000        CON(5)   =SEMI         ;
          0
```

```
2100                    EJECT
2101                ****************************************************************
2102                *
2103                * OPENF  (addr cnt --- t)
2104                *        (addr  cnt --- addr cnt f)
2105                *      This routine will open an fib entry
2106                *      for an already existing file. It will set the USER VAR
2107                *      oSCFIB to reflect the fib# assigned to the file.
2108                *      Returns fail flag: can't find file, wrong type file,
2109                *                         file already open
2110                *      Errors out to system: bad file spec
2111                *
2112                ****************************************************************
2113
2114 E43FF 4D24           CON(5)  =!LOADF
       E
2115 E4404 58     =!OPENF CON(2) #85          OPENF
2116 E4406 F405         NIBASC \OPEN\
       54E4
2117 E440E 6C           CON(2) \F\+#80
2118 E4410 5144 =OPNF   CON(5)  =$OPNF
       E
2119
2120 E4415 8E00 =$OPNF GOSUBL   =SAVEFP         SAVE FORTH POINTERS
       00
2121 E441B 20           P=      0
2122 E441D 143          A=DAT1  A               A(A)=COUNT
2123 E4420 174          D1=D1+  5
2124 E4423 147          C=DAT1  A               C(A)=STRING ADDR
2125 E4426 137          CD1EX                   D1--->STRING
2126 E4429 1B00         D0=(5)  =SCRTCH
       000
2127 E4430 3122         LCASC   \"\             DELIMIT STRING WITH "
2128 E4434 14C          DAT0=C  B
2129 E4437 161  GET01   D0=D0+  2               INCREMENT DEST
2130 E443A CC           A=A-1   A               DECREMENT COUNT
2131 E443C 4F0          GOC     GTDONE          ALL DONE
2132 E443F 14F          C=DAT1  B               READ CHAR
2133 E4442 14C          DAT0=C  B               WRITE CHAR
2134 E4445 171          D1=D1+  2               INCREMENT SOURCE
2135 E4448 6EEF         GOTO    GET01           TRY ANOTHER
2136 E444C 3322 GTDONE LCHEX   FF22            DELIMIT STRING WITH "FF
       FF
2137 E4452 144          DAT0=C  A
2138 E4455 24           P=      4
2139 E4457 8F00         GOSBVL  =R<RSTK         SAVE 5 STACK LEVELS
       000
2140 E445E 1B00         D0=(5)  =SCRTCH         D0--->STRING
       000
2141 E4465 8F00         GOSBVL  =OPENF          OPEN FILE
       000
2142 E446C 4B1          GOC     FOK             NO ERROR
2143
2144                *
2145                * THIS SECTION OF CODE IS FOR AN UNSUCCESSFUL FILE OPEN
```



```
2146                * FOR WHATEVER REASON.
2147                *
2148 E446F 24   FAL     P=      4
2149 E4471 8F00         GOSBVL  =RSTK<R
       000
2150 E4478 8E00         GOSUBL  =GETFP
       00
2151 E447E D0           A=0     A               SET FAILED FLAG
2152 E4480 1C4          D1=D1-  5
2153 E4483 141          DAT1=A  A               PUSH FAILED FLAG ON STACK
2154 E4486 03           RTNCC
2155
2156
2157 E4488 101  FOK     R1=A                    SAVE FIB#
2158 E448B 8F00         GOSBVL  =LOCFIL         GET FILE INFO
       000
2159 E4492 DD           BCEX    A
2160 E4494 136          CD0EX                   GET ADDR OF FIB ENTRY
2161 E4497 164          D0=D0+  5               POINT TO FILE TYPE
2162 E449A D0           A=0     A
2163 E449C 15A3         A=DAT0  4               READ FILE TYPE
2164 E44A0 CC           A=A-1   A               IF LIF1 IT =1
2165 E44A2 CC           A=A-1   A
2166 E44A4 401          GOC     OKFOK           WRONG TYPE
2167 E44A7 111          A=R1
2168 E44AA 8F00         GOSBVL  =CLOSEF         CLOSE FOR BAD TYPE
       000
2169 E44B1 6DBF         GOTO    FAL
2170
2171 E44B5 111  OKFOK   A=R1                    RECOVER FILE TYPE
2172 E44B8 1B16         D0=(5)  =oSCFIB
       BF2
2173 E44BF D2           C=0     A
2174 E44C1 144          DAT0=C  A
2175 E44C4 148          DAT0=A  B               WRITE fib# TO oSCFIB
2176 E44C7 24           P=      4
2177 E44C9 8F00         GOSBVL  =RSTK<R         RESTORE STACK LEVELS
       000
2178 E44D0 8E00         GOSUBL  =GETFP          RESTORE FORTH POINTERS
       00
2179 E44D6 174          D1=D1+  5               POP COUNT OFF STK
2180 E44D9 D0           A=0     A               OVERWRITE ADDR WITH TRUE FLAG
2181 E44DB CC           A=A-1   A               A(A)=TRUE FLAG
2182 E44DD 141          DAT1=A  A               WRITE TRUE FLAG TO DATA STK
2183 E44E0 03           RTNCC                   ALL DONE!
2184
2185            Random  EQU     9
2186            I/Obuf  EQU     10
2187            Iram    EQU     11
2188
2189 E44E2 849  =GETPTR ST=0    Random
2190 E44E5 84A          ST=0    I/Obuf
2191 E44E8 84B          ST=0    Iram
2192 E44EB 8D00         GOVLNG  =GTPTRX
       000
```

```
2193
```



```
2194                    EJECT
2195                ***********************************************
2196                *
2197                * NULL$                          ( -- str )
2198                * Return temporary string of length zero.
2199                *
2200                ***********************************************
2201
2202 E44F2 4044            CON(5)  =!OPENF
          E
2203 E44F7 58      =!NULL$ CON(2)  #85
2204 E44F9 E455            NIBASC  \NULL\
          C4C4
2205 E4501 4A              CON(2)  \$\+#80
2206
2207 E4503 0000    =NULL$  CON(5)  =DOCOL       :
          0
2208 E4508 0000            CON(5)  =LIT
          0
2209 E450D 0500            CON(5)  80
          0
2210 E4512 0000            CON(5)  =PAD         PAD
          0
2211 E4517 0000            CON(5)  =STORE
          0
2212                * EFFECT IS TO MAKE MAXLEN=80, LEN=0
2213 E451C 0000            CON(5)  =PAD         PAD
          0
2214 E4521 0000            CON(5)  =FOUR+       4+
          0
2215 E4526 0000            CON(5)  =ZERO        0
          0
2216 E452B 0000            CON(5)  =SEMI        ;
          0
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER

```
2217                    EJECT
2218                ********************************************
2219                *
2220                * SMOVE                     ( str addr -- )
2221                * Store a string at any given address.
2222                *
2223                ********************************************
2224
2225 E4530 7F44             CON(5)  =!NULL$
           E
2226 E4535 58     =!SMOVE CON(2)  #85
2227 E4537 35D4           NIBASC  \SMOV\
           F465
2228 E453F 5C             CON(2)  \E\+#80
2229
2230 E4541 0000 =SMOVE CON(5)  =DOCOL       :
           0
2231 E4546 0000           CON(5)   =ROT         LEN AD2 AD1
           0
2232 E454B 0000           CON(5)   =2DUP        LEN AD2 AD1 AD2 AD1
           0
2233 E4550 0DF3           CON(5)   =LT          LEN AD2 AD1 (AD2<AD1)
           E
2234 E4555 0000           CON(5)   =ZBRNH       IF moving up
           0
2235 E455A 9100           CON(5)   (SMOV1)-*
           0
2236 E455F 0000           CON(5)   =SWAP        LEN AD1 AD2
           0
2237 E4564 0000           CON(5)   =ROT         AD1 AD2 LEN
           0
2238 E4569 FFA3           CON(5)   =CMOVE       CMOVE
           E
2239 E456E 0000           CON(5)   =SEMI        ELSE moving down
           0
2240                * ELSE CALL CMOV> TO AVOID PROPAGATING COPY
2241 E4573 0000 SMOV1  CON(5)   =SWAP        LEN AD1 AD2
           0
2242 E4578 0000           CON(5)   =ROT         AD1 AD2 LEN
           0
2243 E457D 0000           CON(5)   =CMOV>
           0
2244 E4582 0000           CON(5)   =SEMI        ;
           0
```



```
2245                    EJECT
2246                ********************************************
2247                *
2248                * LEFT$                     ( str pos -- str )
2249                * Return string from start to pos.
2250                *
2251                ********************************************
2252 E4587 5354           CON(5)  =!SMOVE
           E
2253 E458C 58     =!LEFT$ CON(2)  #85
2254 E458E C454           NIBASC  \LEFT\
           6445
2255 E4596 4A             CON(2)  \$\+#80
2256
2257 E4598 0000 =LEFT$ CON(5)  =DOCOL       :
           0
2258 E459D 0000           CON(5)   =ONE         1
           0
2259 E45A2 0000           CON(5)   =SWAP        STR 1 POS
           0
2260 E45A7 0000           CON(5)   =SUB$        get substring from start to pos
           0
2261 E45AC 0000           CON(5)   =SEMI        ;
           0
2262
2263                *
2264                * BASIC
2265                *
2266                * USER VARIABLE
2267                * IN THE NORMAL FORTH ENVIRONMENT IT IS ZERO.
2268                * WHEN EXECUTING FORTHX, IT HAS THE
2269                * RETURN ADDRESS TO THE BASIC KEYWORD.
2270                * THUS IT ALSO HAS THE MEANING OF "IN BASIC?",
2271                * TRUE/FALSE.
2272
2273 E45B1 6B54 =BASIC  CON(5)  =$BASIC
           E
2274 E45B6 8E00 =$BASIC GOSUBL  =DOUSE
           00
2275 E45BC 2ABF           CON(5)  (=oBASIC)
           2
2276 E45C1        =REND3  END
```

```
!*/MOD  Abs  934021 #E4085 -  1851  1868
!2DROP  Abs  931479 #E3697 -   798   816
!>BODY  Abs  930567 #E3307 -   315   333
!?COMP  Abs  932820 #E3BD4 -  1378  1640
!ABORT  Abs  933492 #E3E74 -  1641  1708
!ALOTB  Abs  932701 #E3B5D -  1335  1377
!BEGIN  Abs  932327 #E39E7 -  1169  1206
!BLOCK  Abs  934255 #E416F -  1945  2034
!CMOVE  Abs  932595 #E3AF3 -  1276  1302
!COUNT  Abs  931658 #E374A -   879   902
!DEPTH  Abs  933959 #E4047 -  1828  1850
!DIGIT  Abs  931506 #E36B2 -   817   857
!DOES>  Abs  932255 #E399F -  1144  1168
!ENDOF  Abs  934158 #E410E -  1906  1944
!ENTER  Abs  930650 #E335A -   381   704
!FDROP  Abs  930026 #E30EA -   104   129
!FENCE  Abs  931741 #E379D -   918   945
!FFIND  Abs  930094 #E312E -   130   195
!FIRST  Abs  931889 #E3831 -  1001  1018
!FLUSH  Abs  930475 #E32AB -   287   314
!FSTR$  Abs  929946 #E309A -    63    84
!IOCON  Abs  930353 #E3231 -   226   246
!IOEXP  Abs  930382 #E324E -   247   286
!LASTX  Abs  930608 #E3330 -   334   380
!LEAVE  Abs  931988 #E3894 -  1054  1143
!LEFT$  Abs  935308 #E458C -  2253
!LIMIT  Abs  931922 #E3852 -  1019  1035
!LINE#  Abs  931955 #E3873 -  1036  1053
!LOADF  Abs  934612 #E42D4 -  2035  2114
!M/MOD  Abs  934068 #E40B4 -  1869  1905
!NFILL  Abs  929877 #E3055 -    31    62
!NMOVE  Abs  931336 #E3608 -   735   797
!NULL$  Abs  935159 #E44F7 -  2203  2225
!OKFLG  Abs  931790 #E37CE -   946   967
!ONERR  Abs  931823 #E37EF -   968   983
!OPENF  Abs  934916 #E4404 -  2115  2202
!OVER2  Abs  931295 #E35DF -   705   734
!PLOOP  Abs  932518 #E3AA6 -  1251  1275
!QUERY  Abs  932629 #E3B15 -  1303  1334
!SMOVE  Abs  935221 #E4535 -  2226  2252
!SPACE  Abs  933729 #E3F61 -  1709  1827
!STATE  Abs  931856 #E3810 -   984  1000
!SWAP2  Abs  931612 #E371C -   858   878
!UNTIL  Abs  932414 #E3A3E -  1207  1230
!VARID  Abs  930093 #E30C9 -    85   103
!WHILE  Abs  932481 #E3A81 -  1231  1250
!WIDTH  Abs  931708 #E377C -   903   917
!X<=Y?  Abs  930281 #E31E9 -   196   205
!X=>Y?  Abs  930317 #E320D -   206   225
#TIB    Ext                -  1315
$2DROP  Abs  931496 #E36A8 -   803   801
$<+LP>  Abs  933766 #E3F86 -  1739  1737
$>BODY  Abs  930584 #E3318 -   320   318
$ACTIV  Abs  933470 #E3E5E -  1614  1613
$BASIC  Abs  935350 #E45B6 -  2274  2273
```



```
=$BTFER  Abs  933183 #E3D3F -  1512  1511
 $CMOVE  Abs  932612 #E3B04 -  1281  1280
=$COUNT  Abs  931675 #E375B -   884   882
=$DIGIT  Abs  931523 #E36C3 -   822   820
 $DOLEV  Abs  932117 #E3915 -  1083  1081
=$EMSG   Abs  933231 #E3D6F -  1530  1529
=$ENT    Abs  930707 #E3393 -   400   398
 $FDROP  Abs  930043 #E30FB -   108   107
=$FENCE  Abs  931758 #E37AE -   923   921
=$FILF   Abs  930223 #E31AF -   169   168
=$FIND>  Abs  930995 #E34B3 -   562   560
=$FIRST  Abs  931906 #E3842 -  1006  1004
 $IOCON  Abs  930370 #E3242 -   231   229
 $IOEXP  Abs  930399 #E325F -   252   250
=$LIMIT  Abs  931939 #E3863 -  1024  1022
=$LINE#  Abs  931972 #E3884 -  1041  1039
=$M/MOD  Abs  934085 #E40C5 -  1874  1873
=$NFILL  Abs  929894 #E3066 -    36    34
=$NMOVE  Abs  931353 #E3619 -   740   738
=$OKFLG  Abs  931807 #E37DF -   951   949
=$ONERR  Abs  931840 #E3800 -   972   971
=$OPNF   Abs  934933 #E4415 -  2120  2118
=$OVER2  Abs  931312 #E35F0 -   710   708
 $SP!    Ext                -  1575
=$STATE  Abs  931873 #E3821 -   989   987
=$STERN  Abs  933139 #E3D13 -  1497  1496
=$SWAP2  Abs  931629 #E372D -   863   861
 $UDIV   Ext                -  1874
=$VARID  Abs  930010 #E30DA -    89    88
=$VOC-L  Abs  931774 #E37BE -   935   934
=$WIDTH  Abs  931725 #E378D -   908   906
 $X<=Y?  Abs  930298 #E31FA -   200   199
 $X=>Y?  Abs  930334 #E321E -   210   209
=$XLOOP  Abs  933878 #E3FF6 -  1789  1787
=*/MOD   Abs  934033 #E4091 -  1855
=-FIND   Abs  932864 #E3C00 -  1407
 /       Ext                -  1838
=2DROP   Abs  931491 #E36A3 -   801  1063
 2DUP    Ext                -  2232
=<+LP>   Abs  933761 #E3F81 -  1737  1260
 <;COD   Ext                -  1152
=>BODY   Abs  930579 #E3313 -   318
 >R      Ext                -  1350  1478  1856  1950  2047  2050  2053  2056
=?COMP   Abs  932832 #E3BE0 -  1382  1174  1212  1256  1911
 ?DUP    Ext                -  1435
 ?PAIR   Ext                -  1189  1214  1258  1917
 AB01    Abs  933039 #E3CAF -  1466  1609
 AB04    Abs  932989 #E3C7D -  1443  1474
=ABORT   Abs  933504 #E3E80 -  1645
=ABORTN  Abs  930173 #E317D -   157   149  2071
=ABORTU  Abs  933410 #E3E22 -  1598   150  1067  1386  2072
 ABR00   Abs  933519 #E3E8F -  1648
=ABRT"X  Abs  932999 #E3C87 -  1455  1365
 ABRT0   Abs  933624 #E3EF8 -  1674  1663
 ABRT1   Abs  933524 #E3E94 -  1650
```

```
 ABS     Ext                    -  2041
 ABXX    Abs  933069 #E3CCD     -  1476  1457  1600
=ACTIV   Abs  933465 #E3E59     -  1613
 ADD     Ext                    -  1355  2010  2024
=AGAIN   Abs  932364 #E3A0C     -  1187
 ALLOT   Ext                    -  1341
=ALOTB   Abs  932713 #E3B69     -  1339
 AND     Ext                    -  1661
 AT      Ext                    -   292  1314  1353  1384  1411  1412  1434  1484
                                   1668  1670  1890  1952  1956  1967  1972  1981
                                   1991  2013  2046  2049  2052  2055  2081  2085
 AVMEME  Ext                    -   440   456
 BAD     Abs  932232 #E3988     -  1123  1097
 BAD0    Abs  932242 #E3992     -  1126  1121
 BASEND  Abs  933129 #E3D09     -  1492  1486
=BASIC   Abs  935345 #E45B1     -  2273  1483
=BASIC?  Abs  933089 #E3CE1     -  1482  1471  1648
 BEFLT   Abs  933823 #E3FBF     -  1759  1751  1755
=BEGIN   Abs  932339 #E39F3     -  1173
 BL      Ext                    -  1408  1714
 BLK     Ext                    -  2045  2061  2097
 BLK00   Abs  934322 #E41B2     -  1960  1985
 BLK1    Abs  934467 #E4243     -  1989  1959
 BLK10   Abs  934412 #E420C     -  1978  1963
 BLK2    Abs  934572 #E42AC     -  2016  1995
=BLOCK   Abs  934267 #E417B     -  1949
 BRNCH   Ext                    -  1065  1191  1473  1608  1919
 BSCRIN  Ext                    -  1491
 BSERR   Ext                    -   395
=BTFERR  Abs  933178 #E3D3A     -  1511  1488
 BUFFR   Ext                    -  1968
 C!      Ext                    -   160   298
 C,      Ext                    -  1155
 CAT     Ext                    -  1459  1467  1602  1658  1690  1694  1954  1979
 CLALL   Ext                    -  1427
=CLEAN   Abs  933534 #E3E9E     -  1655  1428
=CLEANX  Abs  932904 #E3C28     -  1423  1443  1647
 CLOSEF  Ext                    -  2168
 CLOSF   Ext                    -  2082
=CM3     Abs  931416 #E3658     -   764   757
 CMOV>   Ext                    -  2243
=CMOVE   Abs  932607 #E3AFF     -  1280  2238
 CMPST-  Ext                    -   200   210
 COMMA   Ext                    -  1078  1158  1194  1219  1263  1923
 COMPL   Ext                    -  1058  1060  1062  1064  1151  1190  1215  1259
                                   1918
 CONT    Abs  933906 #E4012     -  1801  1752  1756
 CONTX   Ext                    -  1410
=COUNT   Abs  931670 #E3756     -   882
 CR      Ext                    -  1685
 CURR    Ext                    -  1669
 DEFIN   Ext                    -  1426
=DEPTH   Abs  933971 #E4053     -  1832
 DIG20   Abs  931561 #E36E9     -   834   847
 DIG25   Abs  931586 #E3702     -   843   849
```



```
 DIG30   Abs  931591 #E3707     -   845   828
 DIG40   Abs  931601 #E3711     -   848   831   838
=DIGIT   Abs  931518 #E36BE     -   820
 DOATN   Ext                    -  1807  1817
 DOCOL   Ext                    -    66   133   157   290   337   385   509  1057
                                  1148  1173  1187  1211  1235  1255  1307  1339
                                  1348  1382  1407  1423  1432  1455  1482  1598
                                  1645  1655  1684  1713  1832  1855  1888  1910
                                  1949  2040  2207  2230  2257
 DODOES  Ext                    -  1157
=DOES>   Abs  932267 #E39AB     -  1148
 DOLEV   Abs  932112 #E3910     -  1081  1066
 DOLEV0  Abs  932156 #E393C     -  1094  1113
 DOLEV1  Abs  932183 #E3957     -  1105  1103
 DOUSE   Ext                    -    89   908   923   935   951   972   989  1006
                                  1024  1041  2274
 DP      Ext                    -  1351  1666
 DROP    Ext                    -   303  1674  1699  1964
 DUP     Ext                    -   158   293  1349  1352  1657  1665  1687  1693
                                  1953  1969  1978  1986  1989  2002  2007
 EMIT    Ext                    -  1695  1715
 EMS01   Abs  933305 #E3DB9     -  1556  1538
 EMS02   Abs  933353 #E3DE9     -  1570  1567
=EMSG    Abs  933226 #E3D6A     -  1529  1470
=ENDOF   Abs  934170 #E411A     -  1910
=ENT     Abs  930702 #E338E     -   398   386
 ENT0    Abs  930765 #E33CD     -   436   421
 ENT01   Abs  930897 #E3451     -   482   488
 ENT02   Abs  930885 #E3445     -   477   474
 ENT1    Abs  930825 #E3409     -   456   448
 ENT2    Abs  930918 #E3466     -   490   466   483
=ENTER   Abs  930662 #E3366     -   385
 EOF     Ext                    -  2075
 ERR     Abs  930682 #E337A     -   390   445
 ERR#    Ext                    -  1497
 ERR1    Abs  930695 #E3387     -   395   392
=ERRTST  Abs  932944 #E3C50     -  1432  1462  1604  1646
 ET0     Abs  932984 #E3C78     -  1440  1437
 EXEC    Ext                    -  1438
 EXPCT   Ext                    -  1309
=F.AND   Abs  930955 #E348B     -   509
 FAIL    Abs  931161 #E3559     -   639   618
 FAL     Abs  935023 #E446F     -  2148  2169
=FENCE   Abs  931753 #E37A9     -   921
 FF1     Abs  930255 #E31CF     -   177   175
 FF2     Abs  930271 #E31DF     -   182   180
=FFIND   Abs  930106 #E313A     -   133
=FILFND  Abs  930218 #E31AA     -   168   143
=FIND>   Abs  930990 #E34AE     -   560  1413
 FINDF   Ext                    -   173
=FIRST   Abs  931901 #E383D     -  1004   291
 FIVE-   Ext                    -  1072  1664
 FLU0    Abs  930517 #E32D5     -   296   302
=FLUSH   Abs  930487 #E32B7     -   290  1429
 FNAME   Ext                    -   142   146
```

FND000	Abs	931023	#E34CF	-	579	596					
FND001	Abs	931051	#E34EB	-	591	611	673				
FND01	Abs	931147	#E354B	-	634	672					
FND1	Abs	931222	#E3596	-	667	676					
FOK	Abs	935046	#E4488	-	2157	2142					
FORSTK	Ext			-	438	457					
FOUR*	Ext			-	2214						
=FROP	Abs	930038	#E30F6	-	107						
=FSTR$	Abs	929958	#E30A6	-	66						
FSX	Abs	930146	#E3162	-	146	141					
FSYNTX	Ext			-	135						
GET01	Abs	934967	#E4437	-	2129	2135					
GETFP	Ext			-	116	178	272	449	500	1117	1123	1574
					1586	2150	2178					
=GETPTR	Abs	935138	#E44E2	-	2189							
GT	Ext			-	1364							
GTDONE	Abs	934988	#E444C	-	2136	2131						
GTPTRX	Ext			-	2192							
GTS13	Ext			-	1570							
HERE	Ext			-	159	161	1069	1175	1192	1217	1261	1264
					1605	1686	1920					
I/OCON	Ext			-	275							
I/OEXP	Ext			-	268							
I/Obuf	Abs	10	#0000A	-	2186	2190						
I>=L	Abs	933812	#E3FB4	-	1755	1750						
=IALLOT	Abs	932733	#E3B7D	-	1348							
IF	Ext			-	1236							
IN	Ext			-	1311	2048	2058	2094				
INTRP	Ext			-	2079							
=IOCON	Abs	930365	#E323D	-	229							
IOEC1	Abs	930455	#E3297	-	275	267						
IOEC2	Abs	930444	#E328C	-	272	279						
IOECO	Abs	930401	#E3261	-	254	233						
=IOEXP	Abs	930394	#E325A	-	250							
IOFAIL	Abs	930465	#E32A1	-	278	269						
Iram	Abs	11	#0000B	-	2187	2191						
=LASTX	Abs	930620	#E333C	-	337							
LATE	Ext			-	1656	1891						
=LEFT$	Abs	935320	#E4598	-	2257							
=LEV	Abs	932000	#E38A0	-	1057							
=LIMIT	Abs	931934	#E385E	-	1022							
=LINE#	Abs	931967	#E387F	-	1039	2017	2054	2064	2088			
LIT	Ext			-	147	338	1073	1153	1156	1357	1659	1836
					1893	1915	1930	1975	2008	2022	2042	2208
=LOADF	Abs	934624	#E42E0	-	2040							
LOCFIL	Ext			-	2158							
LOD0	Abs	934786	#E4382	-	2075	2070						
LOD1	Abs	934811	#E439B	-	2080	2078						
LSTCHR	Abs	931241	#E35A9	-	675	660						
LT	Abs	933840	#E3FD0	-	1766	1749	2233					
M*	Ext			-	1857							
M/	Ext			-	1859							
=M/MOD	Abs	934080	#E40C0	-	1873							
MAT00	Abs	931196	#E357C	-	654	662						
MATCH	Abs	931180	#E356C	-	647	608	631	638				

MATCH1	Abs	931069	#E34FD	-	603	589					
MFERsp	Abs	37901	#0940D	-	9	1585					
MIN	Ext			-	2044						
MINUS	Ext			-	1193	1218	1262	1363	1835	1895	1957	1977
					1982	1993						
=MOVSET	Abs	931430	#E3666	-	774	742						
MVCN1	Abs	931442	#E3672	-	779	775						
NFIL0	Abs	929924	#E3084	-	47	50						
NFIL1	Abs	929939	#E3093	-	52	44						
=NFILL	Abs	929889	#E3061	-	34							
NIB7	Ext			-	400							
=NMOVE	Abs	931348	#E3614	-	738							
NMOVE0	Abs	931359	#E361F	-	742	1284						
=NMOVE1	Abs	931363	#E3623	-	747	754						
NMOVE2	Abs	931388	#E363C	-	755	748						
NMOVE3	Abs	931391	#E363F	-	756	763						
=NULL$	Abs	935171	#E4503	-	2207							
=OKFLG	Abs	931802	#E37DA	-	949							
OKFOK	Abs	935093	#E44B5	-	2171	2166						
ONE	Ext			-	68	164	1176	1188	1213	1607	1672	1973
					2014	2060	2258					
ONE-	Ext			-	1839							
=ONERR	Abs	931835	#E37FB	-	971	1433						
OPENF	Ext			-	2141							
=OPNF	Abs	934928	#E4410	-	2118	2067						
OR	Ext			-	513							
OVER	Ext			-	69	297	1070					
=OVER2	Abs	931307	#E35EB	-	708							
PAD	Ext			-	2210	2213						
PBUF	Ext			-	299	1960						
PFA	Ext			-	1892							
PILERR	Ext			-	387							
=PLOOP	Abs	932530	#E3AB2	-	1255							
POLL	Ext			-	443	1562						
PREV	Ext			-	294	1951	1987	2084				
PSTOR	Ext			-	1367							
QCSP	Ext			-	1149							
=QUERY	Abs	932641	#E3B21	-	1307							
QUIT	Ext			-	1439	1444	1650					
R/W	Ext			-	1974	2015						
R<RSTK	Ext			-	2139							
R>	Ext			-	1059	1061	1354	1476	1858	1889	2016	2087
					2090	2093	2096					
R@	Ext			-	1458	1466	1601	1965	1970	1992	2004	2011
RCL	Ext			-	340							
=REND3	Abs	935361	#E45C1	-	2276							
=ROM;C	Abs	934103	#E40D7	-	1888							
ROT	Ext			-	2231	2237	2242					
RSTK<R	Ext			-	2149	2177						
Random	Abs	9	#00009	-	2185	2189						
S-R0-3	Ext			-	429							
S0	Ext			-	1356	1833						
SAMSGN	Abs	933866	#E3FEA	-	1777	1772						
SAVEFP	Ext			-	108	169	255	1083	1530	1578	2120	
SCFIB	Ext			-	1955	1966	1971	1980	2012	2051	2080	2091

SCRTCH	Ext			-	2126	2140					
SEMI	Ext			-	71	144	165	304	341	388	515	1079
					1159	1177	1195	1220	1240	1266	1317	1342
					1368	1390	1414	1430	1440	1479	1492	1651
					1675	1701	1716	1840	1860	1897	1932	2025
					2099	2216	2239	2244	2261			
=SETERN	Abs	933134	#E3D0E	-	1496	1460	1603					
SETLEV	Ext			-	1265							
SETNAM	Ext			-	134							
SETROM	Abs	931096	#E3518	-	614	594						
SMOV1	Abs	935283	#E4573	-	2241	2235						
=SMOVE	Abs	935233	#E4541	-	2230	163						
SP!	Ext			-	1424	1487						
SP@	Ext			-	1834							
=SPACE	Abs	933741	#E3F6D	-	1713	1700						
SPAN	Ext			-	1313							
=STATE	Abs	931868	#E381C	-	987	1383						
STKDRP	Ext			-	112							
STOP	Abs	933930	#E402A	-	1810	1753	1757					
STORE	Ext			-	295	1071	1076	1312	1316	1667	1671	1896
					1988	2006	2018	2059	2062	2065	2086	2089
					2092	2095	2098	2211				
SUB$	Ext			-	70	2260						
SUC01	Abs	931252	#E35B4	-	681							
SUCC	Abs	930440	#E3288	-	270	276						
SUCESS	Abs	931246	#E35AE	-	678							
SWAP	Ext			-	511	1075	1689	1924	2005	2236	2241	2259
=SWAP2	Abs	931624	#E3728	-	861							
TABLE	Ext			-	625							
THEN	Ext			-	1928							
THREE	Ext			-	1257							
TIB	Ext			-	1308							
TWO	Ext			-	1926							
TWO*	Ext			-	1340							
TWO+	Ext			-	162	1239	1477	1688	1696	1990	2003	
=UNTIL	Abs	932426	#E3A4A	-	1211							
=VARID	Abs	930005	#E30D5	-	88							
=VOC-L	Abs	931769	#E37B9	-	934							
WH01	Abs	933679	#E3F2F	-	1693	1698						
=WHERE	Abs	933634	#E3F02	-	1684							
=WHILE	Abs	932493	#E3A8D	-	1235							
=WIDTH	Abs	931720	#E3788	-	906							
WORD	Ext			-	1409							
=X<=Y?	Abs	930293	#E31F5	-	199							
=X=>Y?	Abs	930329	#E3219	-	209							
XDO	Ext			-	1692							
=XLOOP	Abs	933873	#E3FF1	-	1787	1697						
XLP0	Abs	933933	#E402D	-	1812	1800						
XLP00	Abs	933890	#E4002	-	1795							
XSTR$	Ext			-	67							
XXYY	Ext			-	202	212						
ZBRNH	Ext			-	140	301	1216	1436	1456	1485	1599	1662
					1958	1962	1984	1994	2069	2077	2234	
ZEQ	Ext			-	300	510	512	514	1385	1961	1983	2068
					2076							

ZERO	Ext			-	296	1077	1310	1464	1691	1922	2057	2063
					2083	2215						
eMEM	Ext			-	393							
pENTER	Abs	18	#00012	-	8	444						
pERROR	Ext			-	1563							

```
Saturn Assembler     FT3:_5_CHAR_WORDS              Mon Feb  6, 1984   4:08 pm
Ver. 3.33/Rev. 2241  Statistics                                      Page  84

Input Parameters

  Source file name is MR&FT3

  Listing file name is MR/FT3

  Object file name is MR%FT3::65

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News
```

```
Saturn Assembler     FT4:_SIX_CHAR_WORDS            Fri Jan 13, 1984  10:15 am
Ver. 3.33/Rev. 2241                                                  Page   1

      1                 TITLE  FT4:_SIX_CHAR_WORDS
      2                 RDSYMB MR%GTO
      3                 RDSYMB NZ%SYM
      4 E4600           ABS    #E4600
      5           ************************************************
      6           *
      7           * MR&FT4      <840608.1403>
      8           *
      9           * START OF 6 CHAR WORDS
     10           * ADD ADDITIONAL WORDS AT THIS END
     11           *
     12           ************************************************
```

```
Saturn Assembler     FT4:_SIX_CHAR_WORDS            Fri Jan 13, 1984  10:15 am
Ver. 3.33/Rev. 2241                                                  Page   2

     13                 EJECT
     14           ************************************************
     15           *
     16           * NALLOT  (n --- )  Add n NIBS to the parameter field
     17           *        of the most recently defined word.
     18           *
     19           *        Although its not standard FORTH we have
     20           * changed NALLOT to try and protect users from ruining
     21           * their dictionary and data stack.  Before alloting
     22           * any additional space in the dictionary we make sure
     23           * that the current DP setting allows for the entire
     24           * data stack (40 entries or 200 nibs) and the PAD area
     25           * (90 nibs) plus the # nibs to be alloted.
     26           *
     27           ************************************************
     28
     29 E4600 0000          CON(5)  0               NO MORE 6 CHAR WORDS
              0
     30 E4605 68    =!ALLOT CON(2)  #86             ALLOT
     31 E4607 E414          NIBASC  \NALLO\
              C4C4
              F4
     32 E4611 4D            CON(2)  \T\+#80
     33
     34 E4613 0000 =ALLOT   CON(5)  =DOCOL          :
              0
     35 E4618 0000          CON(5)  =LIT
              0
     36 E461D 93CF          CON(5)  =oALLOT         VECTOR FOR ALLOT
              2
     37 E4622 0000          CON(5)  =AT             CFA TO USE
              0
     38 E4627 0000          CON(5)  =EXEC           EXECUTE
              0
     39 E462C 0000          CON(5)  =SEMI           ;
              0
```

```
                        EJECT
                *****************************************************
                *
                * FACTS ABOUT BASICI/BASICF/BASIC$
                *
                *       Do not confuse them with BASICX; BASICX executes
                * a statement via LINEPARSE: it finishes by returning
                * to the main loop, and it returns nothing to the
                * math stack. These words call the expression-execution
                * subroutine, essentially doing the VAL function to
                * return a number to the math stack.
                *       Control normally returns directly from expression-
                * execution and requires no poll handling as does BASICX.
                * However control will not return in the case of an
                * error or various oddities which can occur with user-defined
                * functions. The main loop- and error-poll handlers recognize
                * that one of these words is in progress by seeing ACTIVE=2.
                *
                *
                ***********************************************
                *
                * BASICI ( str --- )  This word returns the value
                *       of the specified variable(s) to the
                *       FORTH data stack.  The user may also
                *       return the value of a variable expression
                *       (i.e.  " A4*B6/Z4").
                *
                ***********************************************
E4631 5064E             CON(5) =!ALLOT

E4636 68        =!BASI  CON(2) #86
E4638 2414 3594 34      NIBASC \BASIC\
E4642 9C                CON(2) \I\+#80
E4644 9464E     =BASI   CON(5) $BASI

E4649 7F70      $BASI   GOSUB  FLOT             see comments below
E464D 7A31              GOSUB  GETF#
E4651 AFA               A=C    W                VALUE TO A
E4654 7510              GOSUB  FLTOH
E4658 8F00 000          GOSBVL =COLLAP          COLLAPSE MTH STK
                *                               FOR FUTURE BASIC CALLS
E465F 8E00 00           GOSUBL =GETFP
E4665 1C4               D1=D1- 5                PUT 1 ITEM ON STACK
E4668 141               DAT1=A A                WRITE NEW VALUE
E466B 03                RTNCC

E466D 8F00 000  =FLTOH  GOSBVL =FLTDH           TURN TO AN INTEGER
E4674 831               ?XM=0                   OKAY?
```



```
E4677 00                RTNYES
E4679 20                P=      0
E467B 3400 000          LC(5)   =eOVFLW         GIVE OVERFLOW MESSAGE
E4682 8D00 000          GOVLNG =MFERR
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO CONTACT THE MANUFACTURER

```
                        EJECT
                *************************************************
                *
                * BASICF ( str --- )  This word returns the value
                *       of the specified variable(s) to the
                *       floating point stack.  The user may also
                *       return the value of a variable expression
                *       (i.e.  " A4*B6/Z4").
                *
                *************************************************
E4689 6364E     CON(5)  !BASI
E468E 68        =!BASF  CON(2) #86
E4690 241435940 NIBASC \BASIC\
         34
E469A 6C        CON(2) \F\+#80
E469C 1A64E     =BASF   CON(5) $BASF

E46A1 7720      =$BASF  GOSUB  FLOT             see below for details!
E46A5 72E0      GOSUB  GETF#
E46A9 8E00      GOSUBL =STKLFT          GET READY FOR PUSH
         00
E46AF 1B0DBF2   D0=(5) =oX
E46B6 11B       C=R3
E46B9 1547      DAT0=C W                WRITE # TO X REGISTER
E46BD 8F00      GOSBVL =COLLAP          COLLAPSE MTH STK
         000
E46C4 8E00      GOSUBL =GETFP
         00
E46CA 03        RTNCC

E46CC 179   FLOT    D1=D1+ 10           POP STRING
E46CF 8E00      GOSUBL =SAVEFP          SAVE FORTH POINTERS
         00
E46D5 1BC0BF2   D0=(5)  =ACTIVE
E46DC 3420      LC(5)   2           SET ACTIVE TO 2 IN CASE OF
         000
                *                       PAUSE IN USER-DEFINED FUNCTION
E46E3 144       DAT0=C  A
E46E6 1C9       D1=D1- 10             BUT POINT TO STR NOW
E46E9 147       C=DAT1 A              C(A)=CHAR COUNT OF STRING
E46EC 8AE       ?C#0   A              ANY CHARS IN STRING?
E46EF 21        GOYES  FLOT0          YES!
E46F1 20    eIVerr  P=     0
E46F3 3400      LC(5)  =eDATTY        GIVE DATATYPE MF ERROR
         000
E46FA 6D00      GOVLNG =MFERR
         000

E4701 109   FLOT0    R1=C                SAVE IN R1
```



```
E4704 C6        C=C+C  A            C(A)=NIB COUNT OF STRING
E4706 174       D1=D1+ 5
E4709 143       A=DAT1 A            ADDRESS OF STRING
E470C 102       R2=A                SAVE IN R2
E470F 1B00      D1=(5) =MTHSTK
         000
E4716 143       A=DAT1 A            A(A)=TOP OF STACK
E4719 131       D1=A                D1-->TOS
E471C EA        A=A-C  A
E471E 20        P=     0
E4720 3421      LC(5)  18           16 NIBS FOR STRING HEADER
         000
                *                   AND 2 NIBS FOR TRAILING
                *                   CARRIAGE RETURN
E4727 EA        A=A-C  A            TOS-STRING SPACE
E4729 1B00      D0=(5) =AVMEMS
         000
E4730 146       C=DAT0 A            BOTTOM OF STACK
E4733 8B6       ?A>C   A            ENOUGH MEMORY?
E4736 90        GOYES  FL0
E4738 8D00      GOVLNG =MEMERR      REPORT MEMORY ERROR
         000

E473F 111   FL0     A=R1                RECOVER CHAR CNT
E4742 11A       C=R2                RECOVER STRING ADDR
E4745 134       D0=C                D0--->STRING
E4748 CC        A=A-1  A            PRIME COUNTER FOR JUMP TEST

                *
                * NOTE:  THE STRING IS BEING WRITTEN TO THE MATH STACK
                *        REVERSED (JUST THE WAY THEY LIKE 'EM)
                *

E474A 14E   FL1     C=DAT0 B            READ BYTE FROM SOURCE
E474D 161       D0=D0+ 2            POINT TO NEXT BYTE
E4750 1C1       D1=D1- 2
E4753 14D       DAT1=C B            WRITE BYTE TO STACK
E4756 CC        A=A-1  A            DECREMENT COUNTER
E4758 51F       GONC   FL1          GET ANOTHER

E475B 1C1       D1=D1- 2            GET READY FOR CR
E475E 31D0      LC(2)  #0D          CARRIAGE RETURN
E4762 14D       DAT1=C B            WRITE CARRIAGE RETURN TO STACK
E4765 AF2       C=0    W
E4768 1C8       D1=D1- 9
E476B 15D8      DAT1=C 9            WRITE 9 0'S (PART OF HEADER)
E476F 1C4       D1=D1- 5
E4772 111       A=R1
E4775 E4        A=A+1  A            COUNT CARRIAGE RETURN
E4777 C4        A=A+A  A            NIB COUNT
E4779 141       DAT1=A A            WRITE OUT REST OF HEADER
E477C 31F0      LC(2)  #F
E4780 1C1       D1=D1- 2
E4783 14D       DAT1=C B
E4786 85A       ST=1   10            SET FOR VALOO
```

```
185 E4789 01          RTN
186               *
187               * EXAMPLE:  STRING = "HELLO"
188               *
189               * WHEN WE'RE DONE THE MATH STACK LOOKS LIKE THIS:
190               *
191               *  (LOW MEM) F0 A0000 000000000 D0 "OLLEH" (HIGH MEM)
192               *
193 E478B 8F00 GETF#   GOSBVL =VAL00          TURN STRING TO #
          000
194 E4792 84D         ST=0    13            CLEAR PRGM-RUNNING
195 E4795 1537        A=DAT1 W                READ #
196               * CHECK FOR COMPLEX
197 E4799 2F          P=      15
198 E479B 30E         LC(1)   14              CODE FOR COMPLEX
199 E479E 20          P=      0
200 E47A0 814         ASRC
201 E47A3 942         ?C=A    S             DON'T ALLOW IT
202 E47A6 D1          GOYES   eIVgo
203 E47A8 810         ASLC                  RESTORE NUMBER IN REGISTER
204 E47AB 20          P=      0
205               * USE THIS OPPORTUNITY TO RESET ACTIVE TO 1
206 E47AD 1BC0        D0=(5)  =ACTIVE
          BF2
207 E47B4 D2          C=0     A
208 E47B6 E6          C=C+1   A
209 E47B8 144         DAT0=C  A
210 E47BB AF6         C=A     W             GET VALUE
211 E47BE 10B         R3=C                    SAVE IT
212 E47C1 01          RTN
213
214 E47C3 6D2F eIVgo  GOTO    eIVerr
215 E47C7 AC0 =SVSTAT A=0     S
216 E47CA 20          P=      0
217 E47CC 86D         ?ST=0   13            RUNNING
218 E47CF 80          GOYES   SS1
219 E47D1 B44         A=A+1   S
220 E47D4 84D         ST=0    13            CLEAR
221 E47D7 A44  SS1    A=A+A   S
222 E47DA 860         ?ST=0   =NoCont
223 E47DD 80          GOYES   SS2
224 E47DF B44         A=A+1   S
225 E47E2 840         ST=0    =NoCont
226 E47E5 A44  SS2    A=A+A   S
227 E47E8 100         R0=A
228 E47EB 3100        LC(2)   =f1PRGM
229 E47EF 8F00        GOSBVL  =SFLAG?
          000
230 E47F6 110         A=R0
231 E47F9 561         GONC    SS3
232 E47FC B44         A=A+1   S
233 E47FF 100         R0=A
234 E4802 3100        LC(2)   =f1PRGM
235 E4806 8F00        GOSBVL  =SFLAGC
          000
```



```
236 E480D 110         A=R0
237 E4810 A44  SS3    A=A+A   S
238 E4813 3100        LC(2)   =f1SUSP
239 E4817 100         R0=A
240 E481A 8F00        GOSBVL  =SFLAG?
          000
241 E4821 110         A=R0
242 E4824 561         GONC    SS4
243 E4827 B44         A=A+1   S
244 E482A 100         R0=A
245 E482D 3100        LC(2)   =f1SUSP
246 E4831 8F00        GOSBVL  =SFLAGC
          000
247 E4838 110         A=R0
248 E483B 3438 SS4    LC(5)   =oPSTAT
          CF2
249 E4842 136         CD0EX
250 E4845 810         ASLC                  PUT NIBBLE IN A[0]
251 E4848 140         DAT0=A  A
252 E484B 136         CD0EX
253 E484E 01          RTN
```

```
                    EJECT
                *************************************************
                *
                * BASIC$
                *
                * SAME IDEA AS BASICF AND BASIC BUT GETS A STRING
                *
                *************************************************
E4850 E864E     =!BASF  CON(5)  =!BASF
E4855 68        =!BAS$  CON(2)  #86
E4857 241435934 =BASIC$ NIBASC  \BASIC\
E4861 4A                CON(2)  \$\+#80
E4863 00000     =BASIC$ CON(5)  =DOCOL
E4868 BD84E             CON(5)  =B$SUB          LEAVE LEN &ADDR ON STACK
                * NOW MOVE IT TO PAD FROM BASIC MATH STACK
E486D 00006             CON(5)  =DUP            STR LEN LEN
E4872 00000             CON(5)  =>R             STR LEN::LEN
E4877 00000             CON(5)  =PAD
E487C 00000             CON(5)  =TWO+
E4881 00000             CON(5)  =TWO+
E4886 00000             CON(5)  =SWAP           STR PAD+4 LEN
E488B 00000             CON(5)  =CMOVE          PUT STR ON PAD
E4890 00000             CON(5)  =R>             LEN
E4895 00000             CON(5)  =DUP            LEN LEN
E489A 00000             CON(5)  =LIT
E489F 05000             CON(5)  80
E48A4 00000             CON(5)  =MAX          TAKE MAX(LEN,80)
                *
E48A9 00000             CON(5)  =PAD            LEN MAX PAD
E48AE 00000             CON(5)  =C!             STORE MAXLEN
E48B3 00000             CON(5)  =PAD            LEN PAD
E48B8 00000             CON(5)  =TWO+           LEN PAD+2
E48BD 00000             CON(5)  =2DUP           LEN PAD+2 LEN PAD+2
```



```
E48C2 00000             CON(5)  =C!             STORE LEN
E48C7 00000             CON(5)  =TWO+           LEN PAD+4
E48CC 00000             CON(5)  =SWAP
E48D1 08A4E             CON(5)  =CLAP           COLLAPSE MTH STK
E48D6 00000             CON(5)  =SEMI

E48DB 0E84E =B$SUB      CON(5)  =$BSUB
E48E0 78ED  =$BSUB      GOSUB   FLOT
E48E4 7350              GOSUB   VAL0$           D1-->STR,AFTER HEADER
E48E8 84D               ST=0    13            MAKE SURE PRGM RUN FLG CLEAR
E48EB 20                P=      0
E48ED 133               AD1EX
E48F0 131               D1=A
E48F3 1CD               D1=D1-  14              POINT TO LEN
E48F6 AF2               C=0     W
E48F9 147               C=DAT1  A               C=LEN
                * ONLY RETURN 255 CHARS MAX
                * SINCE FORTH STRINGS HAVE 1-BYTE LENGTH FIELD
E48FC D7                D=C     A
E48FE 34EF100           LC(5)   510             255*2
E4905 8B3               ?D>C    A
E4908 40                GOYES   BSU1
E490A DB                C=D     A
E490C 81E   BSU1        CSRB                    NIBS-->CHARS
E490F 109               R1=C
                * RESET ACTIVE TO 1
E4912 1BC0BF2           D0=(5)  =ACTIVE
E4919 D2                C=0     A
E491B E6                C=C+1   A
E491D 144               DAT0=C  A
E4920 8E0000            GOSUBL  =GETFP
E4926 119               C=R1
E4929 1C4               D1=D1-  5
E492C 141               DAT1=A  A               PUT ON STRADDR
E492F 1C4               D1=D1-  5
E4932 145               DAT1=C  A               PUT ON LEN
E4935 03                RTNCC

                =ValSub EQU     10              Indicates VAL is used as a sub
                *
                *
E4937 6B8E      VALERR GOTO     eIVgo
                ***************************************************************
                Return EQU      0
E493B 856       =VAL0$  ST=1    Return
```

```
333 E493E 8F00          GOSBVL =XXHEAD          Remove string header, set up for ST
          000
334 E4945 31D0          LCHEX  0D               Carriage return
335 E4949 8F00          GOSBVL =STKCHR          Append CR to end of string
          000
336 E4950 8F00          GOSBVL =ADHEAD           Put on new string header
          000
337 E4957 8F00          GOSBVL =REVPOP          Reverse string, pop it, C=D0
          000
338 E495E 1B00          D0=(5) =FUNCD1
          000
339 E4965 144           DAT0=C A                Save old D0 in FUNCD1
340 E4968 102           R2=A                    Save string length in R2
341 E496B 8F00          GOSBVL =R<RSTK          Save an RSTK level
          000
342 E4972 8F00          GOSBVL =EXCPAR          Call Execution EXPPAR
          000
343             *       ?ST=0  ValSub           VAL used as a sub?
344             *       GOYES  VAL01            No, then skip
345 E4979 3100          LC(2)  =tEOL
346 E497D 966           ?A#C   B                Is expression followed by CR?
347 E4980 7B            GOYES  VALERR           No, then error
348 E4982 870           ?ST=1  0                Valid expression?
349 E4985 2B            GOYES  VALERR           No, then error
350 E4987 873  VAL01    ?ST=1  3                STRING expression?
351 E498A DA            GOYES  VALERR           No, then error
352 E498C 3100          LC(2)  =tEOL
353 E4990 8F00          GOSBVL =OUTBYT          Append EOL token to expression
          000
354 E4997 DB            C=D    A                C(A)=(MTHSTK)
355 E4999 112           A=R2                    Recall string length
356 E499C C2            C=C+A  A                Add string length
357 E499E 135           D1=C                    D1=(Stack pointer)
358             * Prepare to move code buffer onto stack
359 E49A1 132           AD0EX                   A(A)=End of Source
360 E49A4 11B           C=R3                    C(A)=Start of Source
361 E49A7 8F00          GOSBVL =MOVED2          Move output buffer to stack
          000
362 E49AE 137           CD1EX                   C(A)=Start of dest
363 E49B1 1B00          D0=(5) =MTHSTK
          000
364 E49B8 144           DAT0=C A                Write out pointer to code buffer
365 E49BB 8F00          GOSBVL =RSTK<R
          000
366             * Save two RSTK levels and old D0(program PC) on GOSUB stack
367 E49C2 D2            C=0    A
368 E49C4 3121          LC(2)  18               18 nibbles
369 E49C8 D5            B=C    A                B(A)=Amount memory needs to open up
370 E49CA 22            P=     2
371             *                                Adjust 3 pointers (AVMEME,TFORN,GS
372 E49CC 1A00          D0=(4) =GSBSTK
          00
373             *                               D0 points at last ptr (GSBSTK)
374 E49D2 8F00          GOSBVL =PSHSTK
          000
```



```
375 E49D9 AF2           C=0    W
376 E49DC A7E           C=C-1  W
377 E49DF 1557          DAT1=C W                Write out 3 "F" GOSUB stack entry t
378 E49E3 170           D1=D1+ 1                Skip first entry type
379 E49E6 07            C=RSTK
380 E49E8 145           DAT1=C A                Save an RSTK level
381 E49EB 175           D1=D1+ 6                Point to next entry
382 E49EE 07            C=RSTK
383             *       ?C#0   A                Is this an unused stack level?
384             *       GOYES  VAL10            No, then okay
385             *       C=C+1  A                Yes, then don't leave it zero
386             *VAL10
387 E49F0 145           DAT1=C A                Save another RSTK level
388 E49F3 175           D1=D1+ 6                Point to next entry
389 E49F6 1B00          D0=(5) =FUNCD1
          000
390 E49FD 146           C=DAT0 A                Recall old D0(program PC)
391 E4A00 145           DAT1=C A                Save old D0(program PC)
392 E4A03 1A00          D0=(4) =MTHSTK
          00
393 E4A09 146           C=DAT0 A
394 E4A0C 134           D0=C                    Set PC at start of code buffer
395 E4A0F 135           D1=C                    Set Stack pointer to end AVMEM
396 E4A12 8F00          GOSBVL =EXPR            Evaluate expression
          000
397 E4A19 163           D0=D0+ 4                EOL+18 nibbles on GOSUB stack take
398             *                               20 of nibbles, now move ahead
399             *                               4 more to point where stack item wi
400 E4A1C 136           CD0EX
401 E4A1F 10A           R2=C                    Save address where stack item will
402 E4A22 1B00          D0=(5) =GSBSTK
          000
403 E4A29 142           A=DAT0 A
404 E4A2C 130           D0=A
405 E4A2F 166           D0=D0+ 7                Skip to second entry
406 E4A32 7B30          GOSUB  GETADR
407 E4A36 06            RSTK=C                  Restore an RSTK level
408 E4A38 185           D0=D0- 6                Move back to first entry
409 E4A3B 7230          GOSUB  GETADR
410 E4A3F 06            RSTK=C                  Restore another RSTK level
411 E4A41 16B           D0=D0+ 12               Now move to third entry
412 E4A44 7920          GOSUB  GETADR
413 E4A48 DA            A=C    A
414             *       R3=A                    Save old D0(program PC) in R3
415 E4A4A 164           D0=D0+ 5
416 E4A4D 132           AD0EX                   A(A)=End of save space
417 E4A50 D2            C=0    A
418 E4A52 3121          LC(2)  18               18 nibbles
419 E4A56 EE            C=A-C  A                C(A)=Start of save space
420 E4A58 22            P=     2
421             *                               Adjust 3 pointers(MTHSTK,TFORN,GSBS
422 E4A5A 8F00          GOSBVL =POPSTK          Discard save space on GOSUB stack
          000
423 E4A61 18E           D0=D0- 15                       Point at MTHSTK
424 E4A64 142           A=DAT0 A                Read (MTHSTK)
```

```
425 E4A67 131          D1=A
426 E4A6A 8D00         GOVLNG  =REVPOP
          000
427             * REVPOP RTNS ENDING VALO$
428
429
430 E4A71 146   GETADR C=DAT0 A
431 E4A74 8AE          ?C#0    A
432 E4A77 00           RTNYES
433 E4A79 8D00         GOVLNG =CORUPT
          000
434             *---    LC(2)   =#MMCOR
435             *---    GOTO    =MErr1
436
437             * COLLAPSE MATH STACK SO FUTURE CALLS DON'T HAVE EXTRA STUFF
438 E4A80 58A4 =CLAP   CON(5) =$CLAP
          E
439 E4A85 8E00 =$CLAP  GOSUBL =SAVEFP
          00
440 E4A8B 8F00         GOSBVL =COLLAP
          000
441 E4A92 8E00         GOSUBL =GETFP
          00
442 E4A98 03           RTNCC
```



```
443             EJECT
444             ************************************************
445             *
446             * MAKEBF   ( n --- addr  I/O buff#  true)
447             *          ( n --- false                 )
448             *
449             *       This word finds the next available scratch
450             * I/O buffer number and creates an I/O buffer n nibs
451             * long.  n cannot be > 4095.  If successful, it returns
452             * a true flag, the I/O buffer# and the address of the
453             * beginning of data area in the buffer. If it fails
454             * (n is too large, there are no more free buffer ids,
455             * or not enough memory) it simply returns a false flag.
456             *
457             *
458             * definition changed to make n=nibbles instead of bytes
459             * 12/12/83 B.M.
460             ************************************************************
461 E4A9A 5584             CON(5)  !BAS$
          E
462 E4A9F 68    =!IOMAK  CON(2)  #86
463 E4AA1 D414            NIBASC  \MAKEB\
          B454
          24
464 E4AAB 6C             CON(2)  \F\+#80
465 E4AAD 2BA4 =IOMAKE   CON(5)  $IOMAK
          E
466
467 E4AB2 8E00 $IOMAK    GOSUBL  =SAVEFP
          00
468 E4AB8 143            A=DAT1  A               READ # nibbles
469 E4ABB 34FF           LC(5)   #FFF            MAX SIZE ALLOWED!
          F00
470 E4AC2 8B6            ?A>C    A
471 E4AC5 14             GOYES   FAIL            TOO BIG!
472 E4AC7 DC             ABEX    A               SIZE TO B(A)
473 E4AC9 8F00           GOSBVL  =IOFSCR         GET NEXT FREE I/O #
          000
474 E4AD0 453            GOC     FAIL            DIDN'T WORK
475 E4AD3 D0             A=0     A
476 E4AD5 ABA            A=C     X
477 E4AD8 101            R1=A                    SAVE IN R1
478 E4ADB 8F00           GOSBVL  =I/OALL         ALLOCATE #NIBS FOR THIS ID
          000
479 E4AE2 532            GONC    FAIL            DIDN'T WORK
480 E4AE5 133            AD1EX                   DATA ADDR IN A(A)
481 E4AE8 8E00           GOSUBL  =GETFP          RESTORE FORTH ENVIRONMENT
          00
482 E4AEE 141            DAT1=A  A               WRITE DATA ADDR TO STACK
483 E4AF1 1C4            D1=D1-  5
484 E4AF4 119            C=R1
485 E4AF7 145            DAT1=C  A               WRITE I/O # TO STACK
486 E4AFA D2   SUCC      C=0     A
487 E4AFC CE             C=C-1   A               TRUE FLAG
488 E4AFE 1C4            D1=D1-  5
```

```
489 E4B01 145  MB0      DAT1=C  A            WRITE FLAG TO STACK
490 E4B04 03            RTNCC
491 E4B06 8E00 FAIL     GOSUBL  =GETFP
          00
492 E4B0C D2            C=0     A            FALSE FLAG
493 E4B0E 62FF          GOTO    MB0
```



```
494             EJECT
495             ********************************************
496             *
497             * FINDBF  (n --- addr or 0)  n is the I/O buffer
498             *       ID number.  This routine returns the
499             *       address of the start of data in the
500             *       specified I/O buffer or it returns a
501             *       0 (false flag) if the buffer cannot be
502             *       found.
503             *
504             ********************************************
505 E4B12 F9A4          CON(5)  !IOMAK
          E
506 E4B17 68    =!IOFIN CON(2)  #86
507 E4B19 6494          NIBASC  \FINDB\
          E444
          24
508 E4B23 6C            CON(2)  \F\+#80
509 E4B25 A2B4  =IOFIND CON(5)  $IOFIN
          E
510
511 E4B2A 8E00 $IOFIN   GOSUBL  =SAVEFP       SAVE FORTH ENVIRONMENT
          00
512 E4B30 15F2          C=DAT1  3            READ I/O BUFF NUMBER
513 E4B34 8F00          GOSBVL  =IOFND0      FIND BUFFER
          000
514 E4B3B 5AC           GONC    FAIL         NOT THERE!
515 E4B3E 133           AD1EX                DATA ADDR TO A(A)
516 E4B41 8E00          GOSUBL  =GETFP       RESTORE FORTH ENVIRONMENT
          00
517 E4B47 141  IOFIN0   DAT1=A  A            PUSH ADDR TO STACK
518 E4B4A 03            RTNCC                AND RETURN!
```

```
519                     EJECT
520                 ************************************************
521                 *
522                 * KILLBF  (n --- t/f)  n is the ID# of an I/O buffer
523                 *       to be removed.  If the buffer is there, it
524                 *       is killed and true flag is returned to the
525                 *       user, else a false flag is returned.
526                 *
527                 ************************************************
528 E4B4C 71B4          CON(5)  !IOFIN
        E
529 E4B51 68     =!IOKIL CON(2)  #86
530 E4B53 B494        NIBASC  \KILLB\
        C4C4
        24
531 E4B5D 6C          CON(2)  \F\+#80
532 E4B5F 46B4 =IOKILL CON(5)  $IOKIL
        E
533 E4B64 8E00 $IOKIL  GOSUBL  =SAVEFP      SAVE FORTH ENVIRONMENT
        00
534 E4B6A 15F2         C=DAT1  3            READ ID NUMBER
535 E4B6E 8F00         GOSBVL  =I/ODAL      KILL BUFFER
        000
536 E4B75 509          GONC    FAIL         COULDN'T DO IT!
537 E4B78 8E00 IOK0    GOSUBL  =GETFP       RESTORE FORTH ENVIRONMENT
        00
538 E4B7E D0           A=0     A
539 E4B80 CC           A=A-1   A            true flag!
540 E4B82 44C          GOC     IOFIN0       CARRY ALWAYS CLEAR,FINISH AT SUCC
```



```
541                     EJECT
542
543                 ****************************************************************
544                 *
545                 * UM/MOD    This is the new FORTH 83 name for the old U/
546                 *           routine.
547                 *
548                 * U/  (ud1  u2  --- u3  u4)  Divide double dividend (ud1) by
549                 *       single divisor put remainder (u3) and quotient (u4) onto
550                 *       data stack.
551                 *
552                 ****************************************************************
553 E4B85 15B4          CON(5)   !IOKIL
        E
554 E4B8A 68     =!UDIV  CON(2)  #86             UM/MOD
555 E4B8C 55D4        NIBASC  \UM/MO\
        F2D4
        F4
556 E4B96 4C          CON(2)  \D\+#80
557 E4B98 D9B4 =UDIV   CON(5)  =$UDIV
        E
558
559 E4B9D AF0  =$UDIV  A=0     W            A WILL BE DIVISOR
560 E4BA0 AF2          C=0     W            C         PARTIAL REMAINDERS
561 E4BA3 AF3          D=0     W            D         QUOTIENT
562 E4BA6 BF1          BSL     W
563 E4BA9 BF1          BSL     W            SAVE B[A], CLEAR B[B] FOR DIGIT COUNT
564 E4BAC 143          A=DAT1  A            GET DIVISOR
565 E4BAF 15F9         C=DAT1  10           MSD OF DIVIDEND
566 E4BB3 179          D1=D1+  10
567 E4BB6 147          C=DAT1  A            LSD OF DIVIDEND
568 E4BB9 8AC          ?A#0    A            TEST FOR DIVISOR=0
569 E4BBC D0           GOYES   UDIV1
570 E4BBE D2           C=0     A            SET UP FOR ANSWER
571 E4BC0 272          GONC    UDIV4        B.E.T.
572 E4BC3 A74  UDIV0   A=A+A   W            2*DIVISOR
573 E4BC6 B69          B=B+1   B            INC DIGIT COUNT
574 E4BC9 9FA  UDIV1   ?A<=C   W            MAXIMUM DIGIT YET?
575 E4BCC 7F           GOYES   UDIV0        LOOP IF NOT
576 E4BCE 531          GONC    UDIV3        B.E.T.
577 E4BD1 81C  UDIV2   ASRB                 RIGHT SHIFT DIVISOR
578 E4BD4 A77          D=D+D   W            PREPARE QUOTIENT FOR NEXT TWO-CADE
579 E4BD7 9F6          ?A>C    W            THIS QUOTIENT DIGIT ZERO?
580 E4BDA 80           GOYES   UDIV3        JUMP IF SO
581 E4BDC B72          C=C-A   W            NEXT PARTIAL REMAINDER
582 E4BDF B77          D=D+1   W            QUOTIENT DIGIT IS ONE
583 E4BE2 A6D  UDIV3   B=B-1   B            DEC DIGIT COUNT
584 E4BE5 5BE          GONC    UDIV2        LOOP IF MORE
585 E4BE8 145  UDIV4   DAT1=C  A            WRITE REMAINDER
586 E4BEB AFB          C=D     W            GET QUOTIENT LSD
587 E4BEE 1C4          D1=D1-  5
588 E4BF1 145          DAT1=C  A            WRITE "
589 E4BF4 BF5          BSR     W
590 E4BF7 BF5          BSR     W            RESTORE B[A]
591 E4BFA 03           RTNCC
```

```
592                EJECT
593             ****************************************
594             *
595             * FENTER          41C STACK MANIPULATION
596             *
597             *  essentially does a stack lift.
598             ****************************************
599 E4BFC A8B4          CON(5)  =!UDIV
          E
600 E4C01 68    =!FENTR CON(2)  #86
601 E4C03 6454          NIBASC  \FENTE\         FENTER
          E445
          54
602 E4C0D 2D            CON(2)  \R\+#80
603 E4C0F 41C4  =FENTR  CON(5)  =$FENTR
          E
604 E4C14 8E00  =$FENTR GOSUBL  =SAVEFP
          00
605 E4C1A 8E00          GOSUBL  =STKLFT
          00
606 E4C20 8E00          GOSUBL  =GETFP
          00
607 E4C26 03            RTNCC
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER


```
608                EJECT
609             ****************************************************************
610             *
611             * BASICX  ( str --- )  This routine passes the string
612             *       provided to the BASIC operating system for parsing
613             *       and execution.
614             *       Sets ACTIVE=3 so this state is recognized by
615             *       poll handlers. Control is given up to system -- we
616             *       get it back through the mainloop poll (see BASPOL in
617             *       MR/FTN).
618             *       Command string is put in an I/O-Buffer for LINEP to
619             *       execute from. This buffer is deallocated when we regain
620             *       control.
621             *
622             ****************************************************************
623
624 E4C28 10C4          CON(5)  =!FENTR
          E
625 E4C2D 68    =!BASCX CON(2)  #86
626 E4C2F 2414          NIBASC  \BASIC\
          3594
          34
627 E4C39 8D            CON(2)  \X\+#80
628 E4C3B 0000  =BASICX CON(5)  =DOCOL
          0
629 E4C40 0000          CON(5)  =BASIC      oBASIC VARIABLE
          0
630 E4C45 0000          CON(5)  =AT
          0
631 E4C4A 0000          CON(5)  =ABORTW     IF FROM FORTHX
          0
632 E4C4F F3            CON(2)  =eBASCX
633 E4C51 A6C4          CON(5)  =BASCX
          E
634 E4C56 0000          CON(5)  =ONE
          0
635 E4C5B 0000          CON(5)  =ACTIV
          0
636 E4C60 0000          CON(5)  =STORE
          0
637 E4C65 0000          CON(5)  =SEMI
          0
638
639             ****************************************
640             * SAVE TWO LEVELS OF CPU RTN ON FORTH RTN
641             * WHICH MAY BE TRASHED BY THE BASIC O/S
642             * WHEN ITS EXECUTING THE GIVEN STRING
643
644 E4C6A F6C4  =BASCX  CON(5)  $BASCX
          E
645 E4C6F DC    $BASCX  ABEX    A
646 E4C71 132           AD0EX
647 E4C74 184           D0=D0-  5
648 E4C77 07            C=RSTK
649 E4C79 144           DAT0=C  A
```

```
650 E4C7C 184          D0=D0-  5
651 E4C7F 07           C=RSTK
652 E4C81 144          DAT0=C  A
653 E4C84 132          AD0EX
654 E4C87 DC           ABEX    A           NEW FORTH RTN STK
655 E4C89 8E00         GOSUBL  =SAVEFP
          00
656             *
657             * NOTE:  WHEN WE RETURN (IN MR%FT0) WE MUST GET
658             * RID OF 3 ITEMS ON THE DATA STACK WHICH WE ARE
659             * POPPING NOW BUT WHICH IS NOT REFLECTED LATER SINCE
660             * WE SAVED THE DATA STACK POINTER BEFORE THE POPS.
661             *
662 E4C8F 143          A=DAT1  A           GET STR LEN
663 E4C92 3406         LC(5)   96          MAXIMUM ALLOWED
          000
664 E4C99 8B2          ?A<C    A           IS THE NUMBER OK?
665 E4C9C 40           GOYES   BSX0        YES
666 E4C9E DE           ACEX    A           TAKE MAX OF 96
667 E4CA0 C4   BSX0     A=A+A   A           CHAR-->NIBS
668 E4CA2 3450         LC(5)   5
          000
669 E4CA9 CA           A=C+A   A               ADD 5 FOR LEN(3)&CR
670 E4CAB 101          R1=A                    SAVE LENGTH
671 E4CAE D8           B=A     A               LENGTH FOR I/OALL
672 E4CB0 8F00         GOSBVL  =IOFSCR         GET I/O BUFF # IN C[X]
          000
673 E4CB7 4C0          GOC     nom
674 E4CBA 8F00         GOSBVL  =I/OALL         ALLOCATE IT (LEAVE PTR IN D1)
          000
675 E4CC1 460          GOC     yesmem
676             *
677 E4CC4 60C0 nom      GOTO    nomem
678             *
679             * WE WANT TO SAVE BUFFER ID SO WE CAN KILL IT LATER
680             *
681 E4CC8 D0   yesmem   A=0     A               CLEAR FOR ID
682 E4CCA 816          CSRC                    ID IN C[1-3]-->C[X]
683 E4CCD ABA          A=C     X
684 E4CD0 102          R2=A                    R2=BUFFER ID
685 E4CD3 133          AD1EX                   A=PTR INTO BUFFER
686
687 E4CD6 20           P=      0               set ACTIVE to 3
688 E4CD8 1FC0         D1=(5)  =ACTIVE         which tells rest of FORTH
          BF2
689 E4CDF 3430         LC(5)   3               system that we are doing a
          000
690 E4CE6 145          DAT1=C  A               BASICX stmt.
691
692 E4CE9 8E00         GOSUBL  =GETFP          RESTORE PTRS FOR NMOVE
          00
693
694             * SAVE ID ON FORTH RTN STACK
695
696 E4CEF DC           ABEX    A
```



```
697 E4CF1 132          AD0EX
698 E4CF4 184          D0=D0-  5
699 E4CF7 11A          C=R2                GET ID
700 E4CFA 144          DAT0=C  A           PUT ON RTN STK
701 E4CFD 132          AD0EX
702 E4D00 DC           ABEX    A           PUT PTR BACK IN B
703             *
704             * NOTE: 3 THINGS IN ALL SAVED ON FORTH RTN STACK
705             *
706 E4D02 1BDF         D0=(5)  (=DSTKAD)   WHERE STK PTR SAVED
          AF_
707 E4D09 146          C=DAT0  A
708 E4D0C 137          CD1EX
709 E4D0F 179          D1=D1+  10          DROP STRING
710 E4D12 137          CD1EX
711 E4D15 144          DAT0=C  A
712 E4D18 164          D0=D0+  5           POINT TO RTN STK SAVE AREA
713 E4D1B D9           C=B     A
714 E4D1D 144          DAT0=C  A           STORE NEW RTN TOS
715 E4D20 8E00         GOSUBL  =GETFP
          00
716 E4D26 119          C=R1                GET BACK LENGTH
717 E4D29 CE           C=C-1   A           SUBTRACT 3 FOR LENGTH FIELD
718 E4D2B CE           C=C-1   A
719 E4D2D CE           C=C-1   A
720 E4D2F 109          R1=C                SAVE STR LENGTH
721 E4D32 133          AD1EX               POINT INTO BUFFER
722 E4D35 1553         DAT1=C  X           3 NIB LENGTH FIELD ON STRING
723 E4D39 172          D1=D1+  3           POINT PAST IT
724 E4D3C 133          AD1EX               PTR BACK IN A
725 E4D3F 1C9          D1=D1-  10          BACK UP TO LEAVE STR ADDR ON
726             *                          STACK AND READY FOR DEST ADDR
727 E4D42 141          DAT1=A  A           PUT DEST ADDR ON FORTH STACK
728 E4D45 100          R0=A                SAVE ADDR FOR C IN CALL TO LINEP+
729 E4D48 1C4          D1=D1-  5
730 E4D4B C2           C=C+A   A           POINT TO END OF BUFFER
731 E4D4D CE           C=C-1   A
732 E4D4F CE           C=C-1   A           BACK UP A CHAR
733 E4D51 133          AD1EX               SAVE D1
734 E4D54 135          D1=C
735 E4D57 31D0         LC(2)   #0D         CARRIAGE RTN
736 E4D5B 14D          DAT1=C  B           PUT CR AT END OF LINE
737 E4D5E 131          D1=A                RESTORE D1
738 E4D61 111          A=R1                GET LENGTH AGAIN
739 E4D64 CC           A=A-1   A           SUBTRACT 2 ADDED FOR CR BEFORE
740 E4D66 CC           A=A-1   A           ONLY MOVE THE STRING ITSELF
741 E4D68 141          DAT1=A  A           PUT LENGTH TO MOVE ON STK
742 E4D6B 8E00         GOSUBL  =$NMOVE     PUT STRING IN BUFFER
          00
743 E4D71 118          C=R0                C=LINEPTR FOR LINEP+
744 E4D74 1B00         D0=(5)  =INBS
          000
745 E4D7B 144          DAT0=C  A
746 E4D7E 8D00         GOVLNG  =LINEP*     this entry doesn't do CRLF
          000
```

```
747
748 E4D85 3100 nomem   LC(2)   =eMEM
749 E4D89 8D00        GOVLNG  =MFERR
          000
```



```
750                EJECT
751           ************************************************
752           *
753           *  OUTPUT  ( addr n ---  )  This routine ships an
754           *          arbitrary number of data bytes to a device
755           *          on the HPIL loop specified by the user.
756           *
757           *       n = number of bytes to send
758           *    addr = addr of data to send
759           *
760           *          The device is specified by the contents
761           *          of 3 user variables:  LOOP specifies
762           *          any of 3 loops 1,2,3   PRIME specifies
763           *          the  primary addr of the intended device
764           *          (0 through 31) and SECONDARY specifies
765           *          the secondary device addr (0 thru 31),
766           *          if any, of the intended device.
767           *
768           ************************************************
769
770 E4D90 D2C4         CON(5)  =!BASCX
          E
771 E4D95 68    =!OUTPT CON(2)  #86          OUTPUT
772 E4D97 F455        NIBASC  \OUTPU\
          4505
          55
773 E4DA1 4D          CON(2)  \T\+#80
774
775 E4DA3 0000 =OUTPUT CON(5)  =DOCOL       :
          0
776 E4DA8 DCD4        CON(5)  =OUTP        OUT (WORK HORSE OF RINE)
          E
777 E4DAD 7BD4        CON(5)  =PILERR      ENTER WITH FLAG, IF TRUE
          E
778           *                            DO NOTHING, ELSE GIVE
779           *                            "HPIL ERROR" MESSAGE
780 E4DB2 0000        CON(5)  =SEMI
          0
781
782
783
784 E4DB7 0000 =PILERR CON(5)  =DOCOL
          0
785 E4DBC 0000        CON(5)  =ZEQ         0=
          0
786 E4DC1 0000        CON(5)  =ABORTW      ABORT"
          0
787 E4DC6 A0          CON(2)  =ePILE
788           *       NIBASC  \ :HPIL e\
789           *       NIBASC  \rror\
790 E4DC8 0000        CON(5)  =SEMI
          0
```

```
                EJECT
            ***********************************************
            *
            * OUT : SAME INPUTS AS OUTPUT BUT IT RTNS A
            *       T/F FLAG FOR USE BY ABORT" IN OUTPUT
            *
            ***********************************************

E4DCD 2DD4 =OUTP   CON(5)  $OUT
       E
E4DD2 7390 $OUT    GOSUB  NIB7          (SEE BELOW)

            *
            * STORE 7 NIB FIELD IN STMTR1+2
            *

E4DD6 1B00         D0=(5)  (=STMTR1)+2
       000
E4DDD AF2          C=0     W
E4DE0 15CA         DAT0=C  11           CLEAR STOP CONDITIONS
E4DE4 1586         DAT0=A  7            WRITE 7 NIBS
E4DE8 1B00         D0=(5)  =EOLLEN
       000
E4DEF 15E6         C=DAT0  7            READ EOL LEN & STR
E4DF3 1A00         D0=(4)  (=STMTR0)+11  POSITION TO CKINFO LOCATION
       00
E4DF9 15C6         DAT0=C  7            WRITE OUT EOL INFO
E4DFD 18B          D0=D0-  12           POSITION TO MLFFLG
            *      LC(2)   (=OUTPTt)*16+#F   WE'D LIKE TO DO THIS
            *                                BUT THE ASSEMBLER DOESN'T
            *                                SUPPORT IT, THEREFORE,
            *
E4E00 31F          NIBHEX  31F          THIS CLUES HPIL ITS AN OUTPUT
E4E03 2            CON(1)  OUTPTt     STATEMENT
E4E04 14C          DAT0=C  B            WRITE IT OUT

            *
            * ISSUE SLOW POLL TO HPIL ROM
            *

E4E07 8F00         GOSBVL  =POLL        NEVER CAUSES AN
       000
E4E0E 00           CON(2)  =pPRTCL      ERROR CONDITION
E4E10 831          ?XM=0                WAS POLL HANDLED?
E4E13 04           GOYES   OK

            *
            * POLL WAS NOT HANDLED, RETURN FALSE TO OUTPUT
            *

E4E15 07   BAD     C=RSTK               THROW AWAY DATA STK PTR
E4E17 8E00         GOSUBL  =GETFP       RESTORE FORTH POINTERS
       00
E4E1D D2           C=0     A            FALSE FLAG
```



```

E4E1F 174  BADĨ    D1=D1+  5            THROW AWAY BYTE COUNT
E4E22 145          DAT1=C  A            OVERWRITE ADDR WITH FLAG
E4E25 03           RTNCC

E4E27 1B00 OK1     D0=(5)  (=STMTR0)+1 READ ADDR OF RTNE WHICH
       000
E4E2E 142          A=DAT0  A            WILL SHIP DATA
E4E31 130          D0=A
E4E34 184          D0=D0-  5
E4E37 146          C=DAT0  A            REL OFFSET OF CLEAN UP RTNE
E4E3A 132          AD0EX
E4E3D C2           C=C+A   A            ADDR OF CLEAN UP RTNE
E4E3F 06           RSTK=C               SAVE ADDR ON CPU STK
E4E41 136          CD0EX
E4E44 06           RSTK=C               ADDR OF SEND RTNE ON CPU STK
E4E46 143          A=DAT1  A            READ BYTES TO SEND INTO A(A)
E4E49 174          D1=D1+  5
E4E4C 147          C=DAT1  A            READ ADDR OF BYTES
E4E4F D7           D=C     A            INTO D(A)
E4E51 01           RTN                  GOTO SEND RTNE!
            *                           THIS CODE LEAVES 3 ADDRS
            *                           ON CPU STACK: TOPMOST IS
            *                           SEND RTNE, NEXT IS CLEAN UP
            *                           RTNE, LAST IS RTN TO OK ADDR

E4E53 07   OK      C=RSTK               PTRS TO DATA INTO
E4E55 135          D1=C                 D1
E4E58 7BCF         GOSUB   OK1          SEND DATA
            *                           AND CLEAN UP LOOP
            *
            * RECOVER FORTH POINTERS AND RETURN TRUE FLAG
            *
E4E5C 8E00         GOSUBL  =GETFP
       00
E4E62 D2           C=0     A
E4E64 CE           C=C-1   A            TRUE FLAG
E4E66 48B          GOC     BAD1         B.E.T. TO FINISH IN BAD


E4E69 8E00 =NIB7   GOSUBL  =SAVEFP      SAVE FORTH POINTERS
       00
E4E6F 07           C=RSTK               RTNE RETURN ADDR
E4E71 DE           ACEX  A
E4E73 137          CD1EX
E4E76 06           RSTK=C               SAVE DATA STK POINTER
            *                           ON CPU RTN STK
E4E78 DE           ACEX  A
E4E7A 06           RSTK=C               REPLACE RTN ADDR
            ***********************************************************
            *
            * CREATE 7 NIB ADDR FIELD WHICH SPECIFIES DEVICE
            *
```

```
                * FORMAT:
                *
                *       LLSS SSSP PPPP 0000 LLSS SSSP PPPP
                *
                * WHERE L -- SPECIFIES THE LOOP
                *        S -- SPECIFIES THE SECONDARY ADDR
                *        P -- SPECIFIES THE PRIME ADDR
                *
                ************************************************************

E4E7C 20                P=      0
E4E7E 1B7A              D0=(5)  =oLOOP      PT TO LOOP#
      BF2
E4E85 AF0               A=0     W
E4E88 14A               A=DAT0  B           READ LOOP#
E4E8B A6C               A=A-1   B           DECREMENT 1 TO 0, ETC.
E4E8E BB0               ASL     X
E4E91 A34               A=A+A   X           LEFT SHIFT 5 BITS
E4E94 AF8               B=A     W           COPY PARTIAL TO B(W)
E4E97 AB0               A=0     X
E4E9A 164               D0=D0+  5           PT TO SECONDARY ADDR
E4E9D 14A               A=DAT0  B           READ SECONDARY ADDR
E4EA0 0E6C              B=B!A   B           MERGE SEC. WITH LOOP
E4EA4 164               D0=D0+  5           PT TO PRIME ADDR
E4EA7 14A               A=DAT0  B           READ PRIME ADDR
E4EAA F1                BSL     A
E4EAC C5                B=B+B   A           LEFT SHIFT 5 BITS
E4EAE 0E6C              B=B!A   B           MERGE PRIME WITH ADDR
E4EB2 AF4               A=B     W           COPY TO A
E4EB5 F0                ASL     A           SHIFT A LEFT 4 NIBS
E4EB7 F0                ASL     A           1ST 3 AND LAST 3 NIBS
E4EB9 BF0               ASL     W           ARE DUPLICATES FOR
E4EBC BF0               ASL     W           THIS APPLICATION
E4EBF AB4               A=B     X           DONE!
E4EC2 01                RTN
```



```
                EJECT
                ************************************************
                *
                * SHRINK  (n --- t/f)  Shrink the user's dictionary
                *        space by n nibs, if the user has that many
                *        nibs free.  Return true (-1) flag if okay
                *        or false flag (0) if it failed.
                *
                * changed n to mean nibs instead of bytes
                * 12/12/83 B.M.
                ************************************************

E4EC4 59D4              CON(5) =!OUTPT
      E
E4EC9 68        =!SHRNK CON(2) #86          SHRINK
E4ECB 3584              NIBASC \SHRIN\
      2594
      E4
E4ED5 BC                CON(2) \K\+#80
E4ED7 CDE4      =SHRINK CON(5) $SHRNK
      E

E4EDC 8E00      =$SHRNK GOSUBL =CHK         save FORTH pointers, make
      00
                *                           sure n is legal
E4EE2 06                RSTK=C              save #nibs on CPU rtn stk
E4EE4 1B11              D0=(5) =oS0
      BF2
E4EEB 142               A=DAT0 A            A(A)=top of DATA STK
E4EEE 348C              LC(5)  200          200 FOR DATA STK
      000
E4EF5 EA                A=A-C  A            A(A)=BOTTOM OF DATA STACK
E4EF7 1B39              D0=(5) =oDP
      BF2
E4EFE 348F              LC(5)  248          168 (84 CHARS) REQUIRED BY PAD
      000
                *                           80 NIBS FOR BUFFER BETWEEN PAD
                *                           AND DP
E4F05 D5                B=C    A            SAVE THIS IN B
E4F07 146               C=DAT0 A            C(A)=current dict. ptr.
E4F0A C9                C=C+B  A            EFFECTIVE BEGINNING OF SPACE TO
                *                           SHRINK IN DICTIONARY
E4F0C EE                C=A-C  A            avail space to shrink
E4F0E 540               GONC   SH09         DIFFERENCE IS POSITIVE
E4F11 D2                C=0    A            MAKE SURE WE FAIL COMPARISON
E4F13 D5        SH09    B=C    A            COPY TO B
E4F15 07                C=RSTK              C(A)=desired shrink
E4F17 8B1               ?B>C   A            enough room?
E4F1A 80                GOYES  SH0          yes
E4F1C 8C00              GOLONG =WR1         no, clean up report failure
      00
E4F22 FA        SH0     C=-C   A            2's compl of #nibs
E4F24 06                RSTK=C              save copy
E4F26 D5                B=C    A            B(A)=-1*(#nibs)
```

```
971 E4F28 1B00          D0=(5) =MAINST      A(A)=bottom of DATA STK
          000
972 E4F2F 146           C=DAT0 A            C(A)=addr of FORTHRAM
973
974 E4F32 8F00          GOSBVL =MVMEM+      shrink file, adjust file chain
          000
975 E4F39 8C00          GOLONG =adj         adjust User Variables
          00
```



```
976                     EJECT
977             ***********************************************
978             *
979             * NEGATE  (n  --- -n) leaves 2's comp of number
980             *
981             ***********************************************
982
983 E4F3F 9CE4          CON(5)  !SHRNK
          E
984 E4F44 68    =!NEGAT CON(2)  #86             NEGATE
985 E4F46 E454          NIBASC  \NEGAT\
          7414
          45
986 E4F50 5C            CON(2)  \E\+#80
987 E4F52 75F4  =NEGAT  CON(5)  =$NEGAT
          E
988
989 E4F57 147   =$NEGAT C=DAT1  A               POP n1
990 E4F5A FA            C=-C    A               2'S COMP
991 E4F5C 145           DAT1=C  A               PUSH -n1
992 E4F5F 08            RTNCC
```

```
 993                    EJECT
 994                *****************************************************************
 995                *
 996                * CMOVE> (addr1 addr2 u --- )  C-move-reverse
 997                *        Move u BYTES starting at addr1+n-1 to addr2+n-1
 998                *        and progressing to addr1 to addr2.
 999                *
1000                *        If u=0 do nothing.
1001                *
1002                *        FORTH 83 Standard
1003                *
1004                *****************************************************************
1005
1006 E4F61 44F4          CON(5)  =!NEGAT
           E
1007 E4F66 68    =!CMOV> CON(2)  #86           CMOVE>
1008 E4F68 34D4          NIBASC  \CMOVE\
           F465
           54
1009 E4F72 EB            CON(2)  \>\+#80
1010
1011 E4F74 97F4  =CMOV>  CON(5)  =$CMOV>
           E
1012 E4F79 AF0   =$CMOV> A=0     W            CLEAR FOR SHIFT LATER
1013 E4F7C 143           A=DAT1  A            GET# OF BYTES
1014 E4F7F C4            A=A+A   A            MAKE NIBS
1015 E4F81 6120          GOTO    NMOV0
1016
```



```
1017                    EJECT
1018                *****************************************************************
1019                *
1020                * NMOVE> (addr1  addr2  u  --- )
1021                *
1022                * If u=0 then do nothing, else treat u as an unsigned
1023                * number specifying the number of bytes to move.
1024                * NMOVE> moves n nibbles  from addr1 to addr 2, however,
1025                * it starts moving from (addr1+n-1) to (addr2+n-1).
1026                *
1027                * It uses 1 sublevel, A, C, D0, D1, P, R1, R2
1028                *
1029                *****************************************************************
1030
1031 E4F85 66F4          CON(5)  !CMOV>
           E
1032 E4F8A 68    =!NMOV> CON(2)  #86           NMOVE>
1033 E4F8C E4D4          NIBASC  \NMOVE\
           F465
           54
1034 E4F96 EB            CON(2)  \>\+#80
1035 E4F98 D9F4  =NMOV>  CON(5)  =$NMOV>
           E
1036
1037 E4F9D AF0   =$NMOV> A=0     W            CLEAR FOR SHIFT IN MOVSET
1038 E4FA0 143           A=DAT1  A            GET # NIBS
1039 E4FA3 8E00  NMOV0   GOSUBL  =MOVSET
           00
1040                * If A=0, MOVSET cleans stack and never returns.
1041 E4FA9 810           ASLC                 UNDO SHIFT IN MOVSET
1042 E4FAC 136           CD0EX                GET  ADDR2 IN C
1043 E4FAF C2            C=C+A   A                END OF ADDR2 FIELD
1044 E4FB1 136           CD0EX                    D0=ADDR2+N
1045 E4FB4 137           CD1EX                GET ADDR1
1046 E4FB7 C2            C=C+A   A            ADDR1+N
1047 E4FB9 137           CD1EX
1048 E4FBC 814           ASRC                 A=#OF WORDS
1049 E4FBF 8A8   NMOV1   ?A=0    A            NO MORE WORDS?
1050 E4FC2 61            GOYES   NMOV2
1051 E4FC4 1CF           D1=D1-  16
1052 E4FC7 18F           D0=D0-  16
1053 E4FCA 1577          C=DAT1  W            READ WORD
1054 E4FCE 1547          DAT0=C  W            WRITE IT
1055 E4FD2 CC            A=A-1   A
1056 E4FD4 6AEF          GOTO    NMOV1
1057 E4FD8 810   NMOV2   ASLC
1058 E4FDB 8A8   NMOV3   ?A=0    A            NO MORE NIBS?
1059 E4FDE 61            GOYES   NMOV4
1060 E4FE0 1C0           D1=D1-  1                DECREMENT ADDR1
1061 E4FE3 180           D0=D0-  1                DECREMENT ADDR2
1062 E4FE6 15F0          C=DAT1  1                READ NIBBLE
1063 E4FEA 15C0          DAT0=C  1                WRITE NIBBLE
1064 E4FEE CC            A=A-1   A                DECREMENT COUNTER
1065 E4FF0 6AEF          GOTO    NMOV3            IF NO CARRY DO ANOTHER
1066 E4FF4 8C00  NMOV4   GOLONG  =CM3              FINISH NMOVE
```

```
00
```



```
1067                    EJECT
1068                *************************************
1069                *
1070                * SCFIB   ( --- addr)  returns the
1071                * addr of variable with fib# of
1072                * currently active file
1073                *
1074                *************************************
1075
1076 E4FFA A8F4             CON(5)  !NMOV>
           E
1077 E4FFF 68     =!SCFIB   CON(2)  #86              SCRFIB
1078 E5001 3534             NIBASC  \SCRFI\
           2564
           94
1079 E500B 2C               CON(2)  \B\+#80
1080 E500D 2105   =SCFIB    CON(5)  =$SCFIB
           E
1081
1082
1083 E5012 8E00   =$SCFIB   GOSUBL  =DOUSE
           00
1084 E5018 16BF             CON(5)   (=oSCFIB)
           2
1085
1086 E501D 2205   =BSIZE    CON(5)  =$BSIZE
           E
1087
1088 E5022 8E00   =$BSIZE   GOSUBL =DOUSE
           00
1089 E5028 89BF             CON(5) (=oBSIZE)
           2
```

```
1090                    EJECT
1091            ****************************************************************
1092            *
1093            * CREATE ( -- ) A defining word used to create a dictionary
1094            * entry for <name>, without allocating any parameter field
1095            * memory. When <name> is subsequently executed, the address
1096            * of the first byte of <name>'s parameter field is left on
1097            * the stack.
1098            *
1099            * Form: CREATE <name>
1100            *
1101            ****************************************
1102
1103 E502D FFF4          CON(5)  =!SCFIB
          E
1104 E5032 68    =!CREAT CON(2)  #86             CREATE
1105 E5034 3425          NIBASC  \CREAT\
          5414
          45
1106 E503E 5C            CON(2)  \E\+#80
1107
1108 E5040 0000 =CREAT   CON(5) =DOCOL
          0
1109 E5045 0000          CON(5)  =LIT
          0
1110 E504A 52CF          CON(5)  =oCREAT         VECTOR FOR CREATE
          2
1111 E504F 0000          CON(5)  =AT             GET CFA TO USE
          0
1112 E5054 0000          CON(5)  =EXEC           EXECUTE CONTENTS
          0
1113 E5059 0000          CON(5)  =SEMI           ;
          0
1114
1115            *
1116            * DEFAULT CREATE ROUTINE
1117            *
1118 E505E 0000 =ICREAT CON(5) =DOCOL         :
          0
1119 E5063 5145          CON(5) =LATE          LATEST  get NFA of last defined wor
          E
1120 E5068 0000          CON(5) =COMMA         ,       compile it as a Link Field
          0
1121 E506D 0000 CRE0     CON(5)  =BL           BL      get the new word
          0
1122 E5072 0000          CON(5)  =WORD         WORD    return counted string
          0
1123 E5077 0000          CON(5)  =DUP          DUP     make 2 copies of addr of ne
          0
1124 E507C 0000          CON(5)  =DUP          DUP     word
          0
1125 E5081 0000          CON(5)  =TWO+         2+      point to 1st char of word
          0
1126 E5086 0000          CON(5)  =CAT          C@      get 1st char
          0
```



```
1127 E508B 0000          CON(5)  =ZEQ          0=      if null make it -1 for ABOR
          0
1128
1129 E5090 0000 CRE01    CON(5)  =ABRT"X       ABORT"  if null give message:
          0
1130 E5095 B0            CON(2)  =eRDNL        attempted to redefine null"
1131
1132 E5097 0000 CRE1     CON(5)  =DUP          DUP     dup counted string addr aga
          0
1133 E509C 0000          CON(5)  =CONTX        CONTEXT get addr at which to start
          0
1134 E50A1 0000          CON(5)  =AT           @       our search of current searc
          0
1135 E50A6 0000          CON(5)  =AT           @       vocabulary (CONTEXT)
          0
1136 E50AB 0000          CON(5)  =FIND>        FIND    see if word already exists
          0
1137 E50B0 0000          CON(5)  =ZBRNH        IF      if so,
          0
1138 E50B5 7300          CON(5) (CREA10)-*
          0
1139 E50BA 0000          CON(5)  =2DROP        2DROP   throw away data from FIND
          0
1140 E50BF 0000          CON(5)  =WARN         WARNING see if user want's warning
          0
1141 E50C4 0000          CON(5)  =AT           @
          0
1142 E50C9 0000          CON(5)  =ZBRNH        IF      if warning is set
          0
1143 E50CE E100          CON(5) (CREA10)-*
          0
1144 E50D3 0000          CON(5)  =DUP          DUP     dup addr of counted string
          0
1145 E50D8 0000          CON(5)  =LIT                  get address of rtne
          0
1146 E50DD E3CF          CON(5)  =oUNIQ                which will print
          2
1147 E50E2 0000          CON(5)  =AT                   appropriate "not
          0
1148 E50E7 0000          CON(5)  =EXEC                 unique" message
          0
1149 E50EC 0000 CREA10   CON(5)  =DUP          COPY ADDR FOR REPLACING LEN
          0
1150 E50F1 0000          CON(5)  =CAT          C@      get len of word
          0
1151 E50F6 0000          CON(5)  =WIDTH        WIDTH   get max length allowed
          0
1152 E50FB 0000          CON(5)  =AT           @
          0
1153 E5100 0000          CON(5)  =MIN          MIN     make entry min of two
          0
1154 E5105 0000          CON(5)  =DUP          COPY MIN(LEN,WIDTH)
          0
1155 E510A 0000          CON(5)  =ROT          MIN MIN ADR
          0
```

```
1156 E510F 0000          CON(5)  =C!           REPLACE LEN WITH MIN(LEN,WIDTH)
           0
1157 E5114 0000          CON(5)  =ONE+         1+      add 1 to string len
           0
1158 E5119 0000          CON(5)  =TWO*         2*      get string len in nibs
           0
1159 E511E 3164          CON(5)  =ALLOT        NALLOT  allot space for word
           E
1160 E5123 0000          CON(5)  =DUP          DUP     dup addr of counted string
           0
1161 E5128 0000          CON(5)  =LIT                  again
           0
1162 E512D 0800          CON(5) #80            80      now TOGGLE, set high bit
           0
1163 E5132 1445          CON(5)  =TOGG         TOGGLE  in length byte of counted s
           E
1164 E5137 0000          CON(5)  =HERE         HERE    get addr of next free nib
           0
1165 E513C 0000          CON(5)  =TWO-         2-      point to last char in count
           0
1166 E5141 0000          CON(5)  =LIT                  TOGGLE (set high bit ) in l
           0
1167 E5146 0800          CON(5) #80            80      char of counted string
           0
1168 E514B 1445          CON(5)  =TOGG         TOGGLE
           E
1169 E5150 0000          CON(5)  =LIT                  compile a temporary prologu
           0
1170 E5155 0000          CON(5) =DOVAR         DOVARIABLE     (DOVAR) as Code Fiel
           0
1171 E515A 0000          CON(5)  =COMMA        ,       for this new word
           0
1172 E515F 0000          CON(5)  =CURR         CURRENT store addr of counted strin
           0
1173 E5164 0000          CON(5)  =AT           @       (now its a new word) in the
           0
1174 E5169 0000          CON(5)  =STORE        !       word link field of CURRENT
           0
1175 E516E 0000          CON(5)  =SEMI         ;       vocabulary
           0
1176
1177            * This is the default "not unique" message rtne
1178
1179 E5173 0000 =IUNIQ   CON(5)  =DOCOL
           0
1180 E5178 0000          CON(5)  =COUNT            expects addr of counted string
           0
1181            *                                  gets string count and addr
1182            *                                  suitable for TYPE
1183 E517D 0000          CON(5)  =TYPE             TYPE  string to display device
           0
1184 E5182 0000          CON(5)  =SPACE            SPACE put out 1 delimiting spac
           0
1185 E5187 0000          CON(5)  =PDOTQ            ."    give message
           0
```



```
1186 E518C D0            CON(2) 13                isn't unique "
1187 E518E 9637          NIBASC \isn't \
           E672
           4702
1188 E519A 57E6          NIBASC \unique \
           9617
           5756
           02
1189 E51A8 0000          CON(5)  =SEMI         ;
           0
```

```
1190                    EJECT
1191                ************************************************************
1192                *
1193                * REPEAT   compile time: ( addr1 2 addr2 4 - )   immediate
1194                * compiles BRANCH ( - )  and offset back to BEGIN
1195                *
1196                ************************************************************
1197 E51AD 2305           CON(5)  =!CREAT
          E
1198 E51B2 6C     =!REPET CON(2) #C6              REPEAT
1199 E51B4 2554           NIBASC \REPEA\
          0554
          14
1200 E51BE 4D           CON(2) \T\+#80
1201
1202 E51C0 0000 =REPET  CON(5) =DOCOL
          0
1203 E51C5 0000         CON(5) =?COMP           ensure compile mode
          0
1204 E51CA 0000         CON(5)  =>R             save addr of WHILE
          0
1205 E51CF 0000         CON(5)  =>R             and its code 4
          0
1206 E51D4 0000         CON(5)  =AGAIN          compile BRNCH back to addr1 (BEGIN)
          0
1207 E51D9 0000         CON(5)  =R>             get back addr of WHILE
          0
1208 E51DE 0000         CON(5)  =R>             and the 4
          0
1209 E51E3 0000         CON(5)  =TWO-           make the 4 a 2 for THEN
          0
1210 E51E8 0000         CON(5)  =THEN           fill in HERE at ZBRNH of WHILE
          0
1211 E51ED 0000         CON(5)  =SEMI           ;
          0
```



```
1212                    EJECT
1213                ************************************************************
1214                *
1215                * FORGET  Delete from dictionary <name> which is in the CURRENT
1216                *         vocabulary and all words added to the dictionary after
1217                *         <name> regardless of their vocabulary.
1218                *
1219                * First thing is to check against FENCE (protected dictionary)
1220                * or that word is not in ROM.
1221                * Then forget whole vocabularies which might be after the
1222                * forgotten word. If forgotten word is prior to current vocabular
1223                * make FORTH the CURRENT and CONTEXT vocabulary.
1224                *
1225                * If word is found in current vocabulary, we have to reset the
1226                * last-word-link in the vocabulary; then use VOC-LINK to get the
1227                * most recent vocabulary and chain back through them all (they
1228                * are linked in simple linear fashion) -- resetting the last-word
1229                * in each. We reset the last-word-link by chaining back through
1230                * linked list of words which forms the contents of the vocabulary
1231                *
1232                ************************************************************
1233 E51F2 2B15           CON(5)  =!REPET
          E
1234 E51F7 68     =!FORGT CON(2)  #86             FORGET
1235 E51F9 64F4           NIBASC  \FORGE\
          2574
          54
1236 E5203 4D           CON(2)  \T\+#80
1237
1238 E5205 0000 =FORGT  CON(5)  =DOCOL          :
          0
1239 E520A 0000         CON(5)  =BL
          0
1240 E520F 0000         CON(5)  =WORD             GET WORD FOR FIND
          0
1241 E5214 0000         CON(5)  =CURR
          0
1242 E5219 0000         CON(5)  =AT
          0
1243 E521E 0000         CON(5)  =AT
          0
1244                * LOOK FOR WORD IN CURRENT (& PARENT) VOCABULARIES
1245 E5223 0000         CON(5)  =FIND>            ADDR CH FLAG
          0
1246 E5228 0000         CON(5)  =ZEQ              change sense of FIND> flag
          0
1247 E522D 0000         CON(5)  =ABORTW           abort if not found with
          0
1248 E5232 63           CON(2)  =eNCUR            "not in current vocabulary " mess
1249 E5234 0000         CON(5)  =DROP             DROP char returned by FIND>
          0
1250                *
1251                * Now check that word is not below FENCE or in ROM.
1252                *
1253                * CFA
```

```
1254 E5239 0000          CON(5)  =TWO-           BACK UP BEFORE LAST CHAR
           0
1255 E523E 0000          CON(5)  =ZERO
           0
1256 E5243 0000          CON(5)  =ONE-           -1 (FLAG FOR TRAVERSE)
           0
1257             *         (TRAVERSE WILL FIND 1ST BYTE WITH HIGH BIT SET)
1258 E5248 0000          CON(5)  =TRAVS          FIND NFA
           0
1259 E524D 0000          CON(5)  =DUP            NFA NFA
           0
1260 E5252 0000          CON(5)  =FENCE          get addr of protected area
           0
1261 E5257 0000          CON(5)  =AT             in dictionary
           0
1262 E525C 0000          CON(5)  =U<             word to forget above fence?
           0
1263 E5261 0000          CON(5)  =ABORTW         if not give error
           0
1264 E5266 D0            CON(2)  =ePDIC          "protected dictionary"
1265 E5268 0000          CON(5)  =DUP            NFA NFA
           0
1266             *
1267             * NOTE: we assume here that words in ROM will appear to have
1268             * a negative addr--E0000 and above.
1269             *
1270 E526D 0000          CON(5)  =LTZ            0<  IN ROM?
           0
1271 E5272 0000          CON(5)  =ABORTW         if so abort with same err
           0
1272 E5277 D0            CON(2)  =ePDIC          message
1273 E5279 0000          CON(5)  =>R                 [NFA]
           0
1274 E527E 0000          CON(5)  =R@             NFA  [NFA]
           0
1275 E5283 0000          CON(5)  =CONTX
           0
1276 E5288 0000          CON(5)  =AT             NFA CONTEXT
           0
1277 E528D 0000          CON(5)  =U<
           0
1278             * IF NFA<CONTEXT, THEN MAKE FORTH THE CONTEXT
1279 E5292 0000          CON(5)  =ZBRNH
           0
1280 E5297 9100          CON(5)  (FRGET1)-*
           0
1281 E529C 0000          CON(5)  =LIT
           0
1282 E52A1 7ACF          CON(5)  =FRTHVC
           2
1283 E52A6 0000          CON(5)  =CONTX
           0
1284 E52AB 0000          CON(5)  =STORE
           0
1285             * EQUIVALENT TO TYPING "FORTH" IN FORTH
```



```
1286 E52B0 0000 FRGET1   CON(5)  =R@             NFA
           0
1287 E52B5 0000          CON(5)  =CURR
           0
1288 E52BA 0000          CON(5)  =AT
           0
1289 E52BF 0000          CON(5)  =U<
           0
1290             * IF NFA<CURRENT THEN MAKE FORTH THE DEFINING VOCABULARY
1291 E52C4 0000          CON(5)  =ZBRNH
           0
1292 E52C9 E100          CON(5)  (FRGET2)-*
           0
1293 E52CE 0000          CON(5)  =LIT
           0
1294 E52D3 7ACF          CON(5)  =FRTHVC
           2
1295 E52D8 0000          CON(5)  =CONTX
           0
1296 E52DD 0000          CON(5)  =STORE
           0
1297 E52E2 0000          CON(5)  =DEFIN
           0
1298             * EQUIVALENT TO TYPING "FORTH DEFINITIONS"
1299             *
1300             * Now do the work of actually forgetting things by
1301             * resetting all the pointers.
1302             *
1303 E52E7 0000 FRGET2   CON(5)  =VOC-L
           0
1304 E52EC 0000          CON(5)  =AT             GET LATEST VOC
           0
1305             *
1306             * NEXT LOOP FORGETS VOCABULARIES AFTER FORGOTTEN WORD
1307             * BEGIN **********************   VOC
1308 E52F1 0000 FRGET3   CON(5)  =R@             VOC NFA
           0
1309 E52F6 0000          CON(5)  =OVER           VOC NFA VOC
           0
1310 E52FB 0000          CON(5)  =U<             (VOC AFTER NFA OF FORGETEE?)
           0
1311 E5300 0000          CON(5)  =ZBRNH
           0
1312             * WHILE
1313 E5305 4100          CON(5)  (FRGET4)-*      VOC
           0
1314 E530A 0000          CON(5)  =AT             GET PARENT VOC
           0
1315 E530F 0000          CON(5)  =BRNCH
           0
1316 E5314 DDFF          CON(5)  (FRGET3)-*
           F
1317             * REPEAT
1318             ******************************
1319 E5319 0000 FRGET4   CON(5)  =DUP            VOC VOC
```

```
                0
1320 E531E 0000         CON(5)  =VOC-L
                0
1321 E5323 0000         CON(5)  =STORE          NEW MOST RECENT VOCABULARY
                0
1322            *                               (WHICH IS VOC OF FORGOTTEN WORD)
1323            *
1324            * NEXT LOOP IS FOR EACH VOCABULARY THAT WILL REMAIN
1325            * BEGIN ************************************************
1326 E5328 0000 FRGETL  CON(5)  =DUP            VOC VOC
                0
1327 E532D 0000         CON(5)  =FIVE-          POINT TO WORD-LINK
                0
1328            *                               (LAST WORD IN VOC)
1329            * INNER LOOP TO RESET VOC-LINK OF EACH VOCABULARY
1330            * * * BEGIN
1331 E5332 0000 FRGET5  CON(5)  =AT             POINT TO WORD
                0
1332 E5337 0000         CON(5)  =DUP
                0
1333 E533C 0000         CON(5)  =R@             GET NFA OF FORGOTTEN WORD
                0
1334 E5341 0000         CON(5)  =U<
                0
1335 E5346 0000         CON(5)  =ZEQ
                0
1336            * * * WHILE                     WORD NOT < FORGOTTEN WORD
1337 E534B 0000         CON(5)  =ZBRNH
                0
1338 E5350 4100         CON(5)  (FRGET6)-*
                0
1339 E5355 0000         CON(5)  =FIVE-          POINT TO LINK
                0
1340 E535A 0000         CON(5)  =BRNCH
                0
1341 E535F 3DFF         CON(5)  (FRGET5)-*
                F
1342            * * * REPEAT
1343            * NOW GET PARENT VOCABULARY
1344 E5364 0000 FRGET6  CON(5)  =OVER           VOC LAST-WORD VOC
                0
1345 E5369 0000         CON(5)  =FIVE-          VOC LAST-WORD WORD-LINK-FIELD
                0
1346 E536E 0000         CON(5)  =STORE          SET NEW LAST-WORD-IN-VOC
                0
1347 E5373 0000         CON(5)  =AT             GET PARENT VOCABULARY
                0
1348 E5378 0000         CON(5)  =?DUP           DONE IF 0 LINK
                0
1349 E537D 0000         CON(5)  =ZEQ
                0
1350 E5382 0000         CON(5)  =ZBRNH
                0
1351 E5387 1AFF         CON(5)  (FRGETL)-*
                F
```



```
1352            * UNTIL ************************************************
1353 E538C 0000         CON(5)  =R>             NFA
                0
1354 E5391 0000         CON(5)  =FIVE-          BACK UP PAST LINK
                0
1355 E5396 0000         CON(5)  =DP
                0
1356 E539B 0000         CON(5)  =STORE
                0
1357 E53A0 0000         CON(5)   =SEMI          ;
                0
```

```
1358                    EJECT
1359                ****************************************************************
1360                *
1361                * ?STACK  ( --- ) Issue an error if the stack is out of bounds.
1362                *
1363                ****************************************************************
1364
1365 E53A5 7F15            CON(5)  !FORGT
           E
1366 E53AA 68    =!?STK    CON(2)  #86             ?STACK
1367 E53AC F335            NIBASC  \?STAC\
           4514
           34
1368 E53B6 BC              CON(2)  \K\+#80
1369
1370 E53B8 0000  =?STK     CON(5)  =DOCOL          :
           0
1371 E53BD 0000            CON(5)   =SP@           SP@     current value of stk pointer
           0
1372 E53C2 0000            CON(5)   =S0            S0      bottom of stack
           0
1373 E53C7 0000            CON(5)   =SWAP          SWAP
           0
1374 E53CC 0000            CON(5)   =U<            U<      if current value is > bottom
           0
1375                *                                       (remember stk grows down in m
1376 E53D1 0000            CON(5)   =ABRT"X        ABORT"   then abort with "empty stac
           0
1377 E53D6 E0              CON(2)  =eEMPT                   message
1378                *       NIBASC  \ empty s\
1379                *       NIBASC  \tack\
1380 E53D8 0000            CON(5)   =SP@           SP@     get current stack pointer
           0
1381 E53DD 0000            CON(5)   =PAD                   get PAD addr
           0
1382 E53E2 0000            CON(5)   =LIT
           0
1383 E53E7 8A00            CON(5)  168             we guarantee 84
           0
1384                *                                       bytes between PAD & STK
1385 E53EC 0000            CON(5)   =ADD           +
           0
1386 E53F1 0000            CON(5)   =U<            U<      if current stack pointer is
           0
1387 E53F6 0000            CON(5)   =ABRT"X        ABORT"  < PAD + 168 then abort
           0
1388 E53FB F0              CON(2)  =eFULL                   with "full stack" message
1389                *       NIBASC  \ full st\
1390                *       NIBASC  \ack\
1391 E53FD 0000            CON(5)   =SEMI          ;
           0
```



```
1392                    EJECT
1393                ****************************************************************
1394                *
1395                * LATEST ( --- addr)  Leave the name field addr of the
1396                *         topmost word in the CURRENT vocabulary.
1397                *
1398                ****************************************************************
1399 E5402 AA35              CON(5)  =!?STK
           E
1400 E5407 68    =!LATE     CON(2)  #86             LATEST
1401 E5409 C414            NIBASC  \LATES\
           4554
           35
1402 E5413 4D              CON(2)  \T\+#80
1403
1404 E5415 0000  =LATE     CON(5)  =DOCOL          :
           0
1405 E541A 0000            CON(5)   =CURR           CURRENT
           0
1406 E541F 0000            CON(5)   =AT             @
           0
1407 E5424 0000            CON(5)   =AT             @
           0
1408 E5429 0000            CON(5)   =SEMI           ;
           0
1409
1410
1411                ****************************************************************
1412                *
1413                * TOGGLE  (addr  b --- ) Complement the contents of addr
1414                *          by the bit pattern b.
1415                *
1416                ****************************************************************
1417 E542E 7045              CON(5)  !LATE
           E
1418 E5433 68    =!TOGG     CON(2)  #86             TOGGLE
1419 E5435 45F4            NIBASC  \TOGGL\
           7474
           C4
1420 E543F 5C              CON(2)  \E\+#80
1421
1422 E5441 0000  =TOGG     CON(5)  =DOCOL          :
           0
1423 E5446 0000            CON(5)  =OVER           OVER  addr b addr
           0
1424 E544B 0000            CON(5)  =AT             @     addr b (addr)
           0
1425 E5450 0000            CON(5)  =XOR            XOR   addr XOR[b, (addr)]
           0
1426 E5455 0000            CON(5)  =SWAP           SWAP
           0
1427 E545A 0000            CON(5)  =STORE          !     store XOR result at addr
           0
1428 E545F 0000            CON(5)  =SEMI           ;
           0
```

```
1429                    EJECT
1430                ****************************************************************
1431                *
1432                * NUMBER (addr --- d) Convert the count & char string at addr
1433                *       to a signed 32 bit integer using the current base.
1434                *       If numeric conversion is not possible, an error
1435                *       condition exists.
1436                *
1437                *       (addr ---  ) Floating point number returns nothing!
1438                *       However, it does set DPL to #C0000.
1439                *
1440                *       If the string to be converted has a decimal point
1441                *       in it, it will be interpreted as a floating point
1442                *       number which will be put into the X register after
1443                *       doing a stack lift.
1444                *
1445                *       If the string to be converted has a colon, slash,
1446                *       or comma in it, it will be interpreted as a double
1447                *       number.
1448                *
1449                ****************************************************************
1450 E5464 3345          CON(5)  !TOGG
       E
1451 E5469 68    =!NUMB  CON(2)  #86             NUMBER
1452 E546B E455        NIBASC  \NUMBE\
       D424
       54
1453 E5475 2D          CON(2)  \R\+#80
1454
1455 E5477 0000 =NUMB   CON(5)  =DOCOL          :   (addr1)
       0
1456 E547C 0000        CON(5)  =LIT
       0
1457 E5481 A2CF        CON(5)  =oNUMB           VECTOR FOR NUMBER
       2
1458 E5486 0000        CON(5)  =AT              GET CFA TO USE
       0
1459 E548B 0000        CON(5)  =EXEC            EXECUTE
       0
1460 E5490 0000        CON(5)  =SEMI            ;
       0
1461
1462 E5495 0000 =INUMB  CON(5)  =DOCOL        :
       0
1463 E549A 0000        CON(5)  =QUOTC         COMPILED VERSION OF "
       0
1464 E549F 10          CON(2)  1              MAXLEN&LEN OF STRING "."
1465 E54A1 10          CON(2)  1             (WE'RE LOOKING FOR A
1466 E54A3 E2          CON(2)  \.\                    DECIMAL POINT)
1467            *                             ADDR1 ADDR2 N
1468 E54A5 0000        CON(5)  =ROT           (addr2 n addr1)
       0
1469 E54AA 0000        CON(5)  =DUP           (addr2 n addr1 addr1)
       0
1470 E54AF 0000        CON(5)  =>R            PUSH COPY OF ADDR 1 TO RTN STK
```



```
       0
1471 E54B4 0000        CON(5)  =COUNT         NOW WE HAVE 2 STRINGS
       0
1472 E54B9 0000        CON(5)  =POS           SEE IF THERE'S A DECIMAL POINT
       0
1473 E54BE 0000        CON(5)  =R>            RETRIEVE ADDR1
       0
1474 E54C3 0000        CON(5)  =SWAP          FLAG FROM POS ON TOP
       0
1475 E54C8 0000        CON(5)  =ZBRNH         IF NO MATCH CONTINUE ON
       0
1476 E54CD 8200        CON(5)  (NUMB00)-*
       0
1477
1478                * we have a possible floating point number *
1479
1480 E54D2 0000        CON(5)  =LIT
       0
1481 E54D7 0000        CON(5)  #C0000         F.P. FLAG FOR NUMBER
       C
1482 E54DC 0000        CON(5)  =DPL           STORE IT IN DPL
       0
1483 E54E1 0000        CON(5)  =STORE
       0
1484 E54E6 0000        CON(5)  =COUNT         GET STRING FORM
       0
1485 E54EB C964        CON(5)  =BASF          MAKE IT F.P.# & PUSH TO F.P. STK
       E
1486 E54F0 0000        CON(5)  =SEMI
       0
1487
1488 E54F5 0000 NUMB00  CON(5)   =ZERO          0       addr 0
       0
1489 E54FA 0000        CON(5)   =ZERO          0       addr 0 0
       0
1490 E54FF 0000        CON(5)   =ROT           ROT     0 0 addr
       0
1491 E5504 0000        CON(5)   =DUP           DUP     0 0 addr addr
       0
1492 E5509 0000        CON(5)   =CAT           C@      get count
       0
1493 E550E 0000        CON(5)   =SWAP                  0 0 char-count  addr
       0
1494
1495                *
1496                * THIS NEXT PORTION GETS RID OF ANY LEADING
1497                * BLANKS IN THE STRING
1498                *
1499
1500 E5513 0000 numz    CON(5)   =TWO+          BEGIN 2+  pt to next char
       0
1501 E5518 0000        CON(5)   =DUP                   dup this addr
       0
1502 E551D 0000        CON(5)   =CAT                   get char
       0
```

```
1503 E5522 0000         CON(5)   =LIT                  test for leading
           0
1504 E5527 0200         CON(5)   32                   spaces
           0
1505 E552C 0000         CON(5)   =EQUAL               is it a space?
           0
1506 E5531 0000         CON(5)   =ZBRNH          IF SO THEN
           0
1507 E5536 3200         CON(5)   (numz1)-*
           0
1508 E553B 0000         CON(5)   =SWAP           addr  char-count
           0
1509 E5540 0000         CON(5)   =ONE-           decrement char-count
           0
1510 E5545 0000         CON(5)   =SWAP           char-count  addr
           0
1511 E554A 0000         CON(5)   =ZERO           put 0 on stack for UNTIL test
           0
1512 E554F 0000         CON(5)   =BRNCH
           0
1513 E5554 F000         CON(5)   (numz2)-*
           0
1514 E5559 0000 numz1   CON(5)   =LIT           ELSE -1
           0
1515 E555E FFFF         CON(5)   (0-1)          -1 is for UNTIL test
           F
1516 E5563 0000 numz2   CON(5)   =ZBRNH         THEN UNTIL
           0
1517 E5568 BAFF         CON(5)   (numz)-*
           F
1518 E556D 0000         CON(5)   =TWO-          reset addr to pt to prev char
           0
1519
1520            *
1521            * NOW CHECK TO SEE IF THE FIRST NON-BLANK CHAR IS A MINUS
1522            * SIGN, IF IT IS UPDATE ADDRESS POINTER AND DECREMENT CHAR
1523            * COUNT.  IN ANY EVENT PUT A FLAG ON RETURN STACK WHICH SAYS
1524            * WHETHER NUMBER WAS NEGATIVE (-1) OR POSITIVE (0)
1525            *
1526
1527 E5572 0000         CON(5)   =DUP           char-count  addr  addr
           0
1528 E5577 0000         CON(5)   =TWO+          char-count  addr  addr+2
           0
1529 E557C 0000         CON(5)   =CAT           get char
           0
1530 E5581 0000         CON(5)    =LIT           LIT     compare to
           0
1531 E5586 D200         CON(5)   45             "-"     minus sign
           0
1532 E558B 0000         CON(5)    =EQUAL        =       =?
           0
1533 E5590 0000         CON(5)    =DUP
           0
1534 E5595 0000         CON(5)    =>R
```



```
           0
1535 E559A 0000         CON(5)   =ZBRNH         IF      if neg. increment
           0
1536 E559F 9100         CON(5)   (NUM00)-*              addr to first char
           0
1537 E55A4 0000         CON(5)    =TWO+         2+      beyond minus sign
           0
1538 E55A9 0000         CON(5)   =SWAP
           0
1539 E55AE 0000         CON(5)   =ONE-          decrement char count
           0
1540 E55B3 0000         CON(5)   =SWAP
           0
1541
1542
1543 E55B8 0000 NUM00   CON(5)    =LIT          THEN LIT
           0
1544 E55BD FFFF         CON(5)   (0-1)          -1
           F
1545 E55C2 0000         CON(5)    =DPL          DPL     set decimal pt var. to
           0
1546 E55C7 0000         CON(5)    =STORE        !       true
           0
1547            *
1548            * NOW WE'LL SET UP FOR A DO LOOP WHICH WILL TRY TO CONVERT
1549            * EACH CHAR OF THE STRING INTO A DIGIT ACCORDING TO BASE
1550            *
1551
1552 E55CC 0000         CON(5)   =SWAP
           0
1553 E55D1 0000         CON(5)   =ZERO          loop limit and index
           0
1554 E55D6 0000         CON(5)   =XDO           prepare return stack
           0
1555
1556 E55DB 0000 numy    CON(5)   =TWO+          point to next char
           0
1557 E55E0 0000         CON(5)   =DUP
           0
1558 E55E5 0000         CON(5)   =>R            save copy of addr on rtn stk
           0
1559 E55EA 0000         CON(5)   =CAT           get char
           0
1560 E55EF 0000         CON(5)   =BASE
           0
1561 E55F4 0000         CON(5)   =AT
           0
1562 E55F9 0000         CON(5)   =DIGIT
           0
1563
1564 E55FE 0000         CON(5)   =ZBRNH         IF successful
           0
1565 E5603 9600         CON(5)   (numy0)-*
           0
1566 E5608 0000         CON(5)   =SWAP
```

```
          0
1567 E560D 0000          CON(5)  =BASE
          0
1568 E5612 0000          CON(5)  =AT
          0
1569 E5617 0000          CON(5)  =U*
          0
1570 E561C 0000          CON(5)  =DROP
          0
1571 E5621 0000          CON(5)  =ROT
          0
1572 E5626 0000          CON(5)  =BASE
          0
1573 E562B 0000          CON(5)  =AT
          0
1574 E5630 0000          CON(5)  =U*
          0
1575 E5635 0000          CON(5)  =D+
          0
1576 E563A 0000          CON(5)  =DPL
          0
1577 E563F 0000          CON(5)  =AT
          0
1578 E5644 0000          CON(5)  =ONE+
          0
1579
1580 E5649 0000          CON(5)  =ZBRNH          IF------+
          0
1581 E564E 4100          CON(5)  (numy1)-*               |
          0
1582 E5653 0000          CON(5)  =ONE                    |
          0
1583 E5658 0000          CON(5)  =DPL                    |
          0
1584 E565D 0000          CON(5)  =PSTOR          +!      |
          0
1585            *                                       |
1586 E5662 0000 numy1    CON(5)  =BRNCH          THEN----+
          0
1587 E5667 6B00          CON(5)  (numy2)-*
          0
1588
1589            *
1590            * DIGIT failed.  If the char was a blank we
1591            * stop converting.  If the char was a ","  "/"
1592            * or a ":" we try to convert some more.  If its
1593            * anything else we bomb out with an error.
1594            *
1595
1596 E566C 0000 numy0    CON(5)  =R@             ELSE R@
          0
1597 E5671 0000          CON(5)  =CAT
          0
1598 E5676 0000          CON(5)  =DUP
          0
```



```
1599 E567B 0000          CON(5)  =LIT
          0
1600 E5680 0200          CON(5)  32              SPACE
          0
1601 E5685 0000          CON(5)  =EQUAL          test for space
          0
1602
1603 E568A 0000          CON(5)  =ZBRNH          IF a space
          0
1604 E568F 8200          CON(5)  (numy3)-*
          0
1605 E5694 0000          CON(5)  =R>
          0
1606 E5699 0000          CON(5)  =2DROP
          0
1607 E569E 0000          CON(5)  =R>
          0
1608 E56A3 0000          CON(5)  =R>
          0
1609 E56A8 0000          CON(5)  =2DROP
          0
1610 E56AD 0000          CON(5)  =BRNCH
          0
1611 E56B2 F700          CON(5)  (numy22)-*
          0
1612 E56B7 0000 numy3    CON(5)  =DUP            ELSE DUP
          0
1613 E56BC 0000          CON(5)   =LIT           LIT     dup this char
          0
1614 E56C1 C200          CON(5)  44              ","     is it a comma?  (c c , )
          0
1615 E56C6 0000          CON(5)   =EQUAL         =       (c f)
          0
1616 E56CB 0000          CON(5)   =SWAP                  (f c)
          0
1617 E56D0 0000          CON(5)   =DUP                   (f c c)
          0
1618 E56D5 0000          CON(5)   =LIT
          0
1619 E56DA F200          CON(5)   47             is it a slash?  (f c c / )
          0
1620 E56DF 0000          CON(5)   =EQUAL                 (f c f)
          0
1621 E56E4 0000          CON(5)   =SWAP                  (f f c)
          0
1622 E56E9 0000          CON(5)   =LIT
          0
1623 E56EE A300          CON(5)   58             is it a colon?  (f f c : )
          0
1624 E56F3 0000          CON(5)   =EQUAL                 (f f f)
          0
1625 E56F8 0000          CON(5)   =OR                    (f  (f OR f) )
          0
1626 E56FD 0000          CON(5)   =OR                    (f OR f OR f)
          0
```

```
1627 E5702 0000          CON(5)   =ZEQ          0=      IF NOT ONE OF THESE
           0
1628
1629 E5707 0000          CON(5)   =ABORTW        ABORT"   not recognized
           0
1630 E570C 01            CON(2)   =eNREC
1631
1632 E570E 0000          CON(5)   =ZERO          0
           0
1633 E5713 0000          CON(5)   =DPL           DPL
           0
1634 E5718 0000          CON(5)   =STORE         !
           0
1635 E571D 0000 numy2    CON(5)   =R>            THEN THEN R>
           0
1636
1637 E5722 0000          CON(5)   =XLOOP         LOOP
           0
1638 E5727 4BEF          CON(5)   (numy)-*
           F
1639 E572C 0000          CON(5)   =DROP
           0
1640 E5731 0000 numy22   CON(5)   =R>            R>
           0
1641
1642           *
1643           * RECOVER SIGN FLAG FROM RETURN STACK
1644           * IF FLAG IS TRUE NUMBER WAS NEGATIVE
1645           *
1646
1647 E5736 0000          CON(5)   =ZBRNH         IF
           0
1648 E573B A000          CON(5)   (NUM1)-*
           0
1649 E5740 0000          CON(5)   =DNGAT         DNEGATE
           0
1650 E5745 0000 NUM1     CON(5)   =SEMI          THEN ;
           0
```



```
1651                     EJECT
1652           ***#######################################
1653           *
1654           * ABORT"  (flag --- )  If flag is true
1655           *         give error message that follows
1656           *         ABORT" and run ABORT sequence
1657           *         otherwise skip message.
1658           *
1659           ***#######################################
1660 E574A 9645          CON(5)   !NUMB
           E
1661 E574F 6C    =!ABRT" CON(2)   #C6             ABORT"
1662 E5751 1424          NIBASC   \ABORT\
           F425
           45
1663 E575B 2A            CON(2)   \"\+#80
1664
1665 E575D 0000 =ABRT"   CON(5)   =DOCOL         :
           0
1666
1667 E5762 0000          CON(5)   =DUP           DUP FLAG
           0
1668 E5767 0000          CON(5)   =ZBRNH         SKIP ON ERROR TEST IF 0
           0
1669 E576C A000          CON(5)   (ABO)-*
           0
1670 E5771 0000          CON(5)   =ERRTST        TEST FOR (& DO )ON ERROR RTNE
           0
1671
1672 E5776 0000 ABO      CON(5)   =STATE
           0
1673 E577B 0000          CON(5)   =AT            COMPILING?
           0
1674 E5780 0000          CON(5)   =ZBRNH         IF compiling then
           0
1675 E5785 F000          CON(5)   (AB5)-*        compile in runtime
           0
1676 E578A 0000          CON(5)   =COMPL         code for ABORT"
           0
1677 E578F E085          CON(5)   =ABRTC
           E
1678 E5794 0000 AB5      CON(5)   =LIT           THEN
           0
1679 E5799 2200          CON(5)   34             "  if string does not
           0
1680 E579E 3095          CON(5)   =GCHAR            end with " then give
           E
1681 E57A3 0000          CON(5)   =ABRT"X        'no ending " 'message
           0
1682 E57A8 50            CON(2)   =eNO"          GCHAR also moves string to
1683 E57AA 0000          CON(5)   =STATE         HERE
           0
1684 E57AF 0000          CON(5)   =AT            COMPILING?
           0
1685 E57B4 0000          CON(5)   =ZBRNH         IF compiling then
```

```
          0
1686 E57B9 F000          CON(5)  (AB6)-*
          0
1687 E57BE 4495          CON(5)  =MOVDP          MOVE DICTIONARY PTR
          E
1688 E57C3 0000          CON(5)  =SEMI           ELSE   (cheat and end here)
          0
1689
1690 E57C8 0000 AB6      CON(5)   =SWAP          GET ORIGINAL FLAG, LEAVE STR ADDR
          0
1691 E57CD 0000          CON(5)   =ZBRNH           IF   true give  error
          0
1692 E57D2 2300          CON(5)  (AB0)-*
          0
1693 E57D7 0000 AB3      CON(5)   =COUNT
          0
1694 E57DC 0000          CON(5)   =TYPE            TYPE     with desired string.
          0
1695 E57E1 0000          CON(5)   =CLALL           close all files
          0
1696 E57E6 0000          CON(5)   =SP!             reset data stk pointer
          0
1697 E57EB E9CF          CON(5)   (=FRTHHD)+12
          2
1698 E57F0 0000          CON(5)   =DEFIN           FORTH DEFINITIONS
          0
1699 E57F5 0000          CON(5)   =CLEAN                   get rid of garbage in d
          0
1700 E57FA 0000          CON(5)   =BASIC?
          0
1701 E57FF 0000          CON(5)   =QUIT            QUIT  we never return from QUIT
          0
1702
1703 E5804 0000 AB0      CON(5)   =DROP           DROP ADDR
          0
1704 E5809 0000          CON(5)   =SEMI
          0
1705
1706
1707
1708
1709            ******************************
1710            *
1711            * Run time rtne for ABORT"
1712            *
1713            ******************************
1714
1715 E580E 0000 =ABRTC   CON(5)   =DOCOL
          0
1716 E5813 0000          CON(5)   =DUP            DUP FLAG
          0
1717 E5818 0000          CON(5)   =ZBRNH          IF NON-ZERO
          0
1718 E581D A000          CON(5)   (ABER)-*        TEST FOR (& DO )
          0
```



```
1719 E5822 0000          CON(5)  =ERRTST         ON ERROR RTNE
          0
1720 E5827 0000 ABER     CON(5)  =R@              LEAVE FOR CODE AT AB3
          0
1721 E582C 0000          CON(5)  =SWAP            SWAP FLAG & ADDR
          0
1722            * NOW FIX RETURN TO POINT AFTER MESSAGE
1723 E5831 0000          CON(5)  =R>
          0
1724 E5836 0000          CON(5)  =DUP
          0
1725 E583B 0000          CON(5)  =CAT
          0
1726 E5840 0000          CON(5)  =ONE+
          0
1727 E5845 0000          CON(5)  =TWO*
          0
1728 E584A 0000          CON(5)  =ADD
          0
1729 E584F 0000          CON(5)  =>R
          0
1730 E5854 0000          CON(5)  =ZEQ           IF FLAG NONZERO, GO ON
          0
1731 E5859 0000          CON(5)  =ZBRNH
          0
1732 E585E 97FF          CON(5)  (AB3)-*
          F
1733 E5863 0000          CON(5)  =DROP
          0
1734 E5868 0000          CON(5)  =SEMI
          0
1735
```

```
1736                    EJECT
1737                ************************************************************
1738                *
1739                * SPACES (n --- ) Transmit N spaces to current output device
1740                *       takes no action for n <=0
1741                *
1742                ************************************************************
1743 E586D F475           CON(5)  !ABRT"
          E
1744 E5872 68    =!SPACS  CON(2)  #86             SPACES
1745 E5874 3505          NIBASC  \SPACE\
          1434
          54
1746 E587E 3D            CON(2)  \S\+#80
1747
1748 E5880 0000  =SPACS  CON(5)  =DOCOL          :
          0
1749 E5885 0000          CON(5)   =ZERO          0
          0
1750 E588A 0000          CON(5)   =MAX           MAX
          0
1751 E588F 0000          CON(5)   =?DUP          ?DUP
          0
1752 E5894 0000          CON(5)   =ZBRNH         IF
          0
1753 E5899 E100          CON(5)  (SPC0)-*
          0
1754 E589E 0000          CON(5)   =ZERO          0
          0
1755 E58A3 0000          CON(5)   =XDO           DO
          0
1756 E58A8 0000  SPC1    CON(5)   =SPACE         SPACE
          0
1757 E58AD 0000          CON(5)   =XLOOP         LOOP
          0
1758 E58B2 6FFF          CON(5)  (SPC1)-*
          F
1759 E58B7 0000  SPC0    CON(5)   =SEMI          ;
          0
```



```
1760                    EJECT
1761                ************************************************************
1762                *
1763                * SMUDGE  ( --- )  Toggle's the "smudge bit" in a definition's
1764                *         name field.  This prevents an uncompleted definition
1765                *         from being found during dictionary searches.
1766                *
1767                ************************************************************
1768 E58BC 2785           CON(5)  =!SPACS
          E
1769 E58C1 68    =!SMUDG  CON(2)  #86             SMUDGE
1770 E58C3 35D4          NIBASC  \SMUDG\
          5544
          74
1771 E58CD 5C            CON(2)  \E\+#80
1772
1773 E58CF 0000  =SMUDG  CON(5)  =DOCOL          :
          0
1774 E58D4 5145          CON(5)   =LATE          LATEST
          E
1775 E58D9 0000          CON(5)   =LIT
          0
1776 E58DE 0200          CON(5)  #20             20 HEX
          0
1777 E58E3 1445          CON(5)   =TOGG          TOGGLE
          E
1778 E58E8 0000          CON(5)   =SEMI          ;
          0
1779
1780                ******************************************************
1781                *
1782                * ?PAIRS   (n1 n2 --- )  issues error message if n1#n2
1783                *
1784                ******************************************************
1785
1786 E58ED 0000  =?PAIR  CON(5)  =DOCOL          :
          0
1787 E58F2 0000          CON(5)   =MINUS         -
          0
1788 E58F7 0000          CON(5)   =ABORTW        ABORT"
          0
1789 E58FC 11            CON(2)  =eCOND
1790                *       NIBASC  \ conditi\
1791                *       NIBASC  \onals no\
1792                *       NIBASC  \t paired\
1793 E58FE 0000          CON(5)   =SEMI          ;
          0
1794                ****************************************
1795                *
1796                * GCHAR  ( c -- addr t/f)  moves string to HERE
1797                * looking for c; returns true if no ending c
1798                * otherwise false; also returns addr of string
1799                *
1800                ****************************************
1801
```

```
1802 E5903 0000 =GCHAR CON(5)  =DOCOL      :
           0
1803 E5908 0000        CON(5)  =DUP        DUP  (c c )
           0
1804 E590D 0000        CON(5)  =WORD            (c addr)
           0
1805 E5912 0000        CON(5)  =DUP             (c addr addr)
           0
1806 E5917 0000        CON(5)  =CAT        C@
           0
1807 E591C 0000        CON(5)  =ONE+       1+
           0
1808 E5921 0000        CON(5)  =TWO*       2*
           0
1809 E5926 0000        CON(5)  =OVER
           0
1810 E592B 0000        CON(5)  =ADD        +
           0
1811 E5930 0000        CON(5)  =CAT        C@   get last-char
           0
1812 E5935 0000        CON(5)  =ROT             ( addr  last-char  c)
           0
1813 E593A 0000        CON(5)  =<>         true if chars <>
           0
1814 E593F 0000        CON(5)  =SEMI       ;
           0
```



```
1815                EJECT
1816           ********************************************
1817           *
1818           * MOVEDP:
1819           *
1820           * MOVE DICTIONARY POINTER PAST STRING AT END
1821           * OF DICTIONARY (PUT THERE BY WORD)
1822           *
1823           ********************************************
1824
1825 E5944 0000 =MOVDP CON(5)  =DOCOL      :
           0
1826 E5949 0000        CON(5)  =CAT        C@
           0
1827 E594E 0000        CON(5)  =ONE+       1+
           0
1828 E5953 0000        CON(5)  =TWO*       2*
           0
1829 E5958 3164        CON(5)  =ALLOT      NALLOT
           E
1830 E595D 0000        CON(5)  =SEMI       ;
           0
1831           ********************************************************
1832           *
1833           * CLOSEF  ( fib# --- )  This routine will close the file
1834           *         identified by the fib# on the data stack.
1835           *
1836           *
1837           ********************************************************
1838
1839 E5962 1C85            CON(5)  =!SMUDG
           E
1840 E5967 68    =!CLOSF CON(2) #86           CLOSEF
1841 E5969 34C4          NIBASC \CLOSE\
           F435
           54
1842 E5973 6C            CON(2) \F\+#80
1843 E5975 A795 =CLOSF   CON(5)  =$CLOSF
           E
1844
1845 E597A 143   =$CLOSF A=DAT1  A
1846 E597D 174           D1=D1+  5              POP fib# FROM DATA STK
1847 E5980 8E00          GOSUBL  =SAVEFP        SAVE FORTH POINTERS
           00
1848 E5986 8F00          GOSBVL  =CLOSEF        CLOSE THIS FILE
           000
1849 E598D 8C00          GOLONG  =GETFP         RESTORE FORTH POINTERS
           00
```

```
1850                EJECT
1851            ************************************************************
1852            *
1853            * BUFFER  (n1 n2 --- addr)  This routine will obtain the
1854            *         next block buffer, assigning it to block n.  The
1855            *         block is not read from mass storage.
1856            *
1857            * n1  --- desired line #
1858            * n2  --- fib# identifying the desired file
1859            * addr -- addr of 1st data byte in the buffer
1860            *
1861            ************************************************************
1862
1863
1864 E5993 0000 =BUFFR CON(5)  =DOCOL        :
               0
1865 E5998 0000        CON(5)    =USE         USE     get addr of next
               0
1866 E599D 0000        CON(5)    =AT          @       buffer to use and
               0
1867 E59A2 0000        CON(5)    =DUP         DUP     duplicate it, put
               0
1868 E59A7 0000        CON(5)    =>R          >R      a copy on the return stk,
               0
1869 E59AC 0000 BUF0   CON(5)    =PBUF        BEGIN +BUF   get addr of next
               0
1870 E59B1 0000        CON(5)    =ZBRNH       UNTIL        buffer to use &
               0
1871 E59B6 6FFF        CON(5)   (BUF0)-*
               F
1872 E59BB 0000        CON(5)    =USE         USE       store it in
               0
1873 E59C0 0000        CON(5)    =STORE       !         USE
               0
1874 E59C5 0000        CON(5)    =R@          R@        get buffer addr
               0
1875 E59CA 0000        CON(5)    =C!          C!        store new fib# into it
               0
1876 E59CF 0000        CON(5)    =R@          R@        get buffer addr
               0
1877 E59D4 0000        CON(5)    =TWO+        2+        make it line# addr
               0
1878 E59D9 0000        CON(5)    =STORE       !         store new line# into it
               0
1879 E59DE 0000        CON(5)    =R@          R@        get buffer addr
               0
1880 E59E3 0000        CON(5)    =PREV        PREV      mark it as previously us
               0
1881 E59E8 0000        CON(5)    =STORE       !         buffer
               0
1882 E59ED 0000        CON(5)    =R>          R>        get buffer addr off rtn
               0
1883 E59F2 0000        CON(5)    =LIT
               0
1884 E59F7 B000        CON(5)   11            11
```



```
               0
1885 E59FC 0000        CON(5)    =ADD         +         make it addr of buffer d
               0
1886 E5A01 0000        CON(5)    =LIT                   we've just assigned, or
               0
1887 E5A06 FFFF        CON(5)    #FFFF        -1        more importantly, reassi
               0
1888 E5A0B 0000        CON(5)    =LINE#       LINE#     a buff to a file, we mus
               0
1889 E5A10 0000        CON(5)    =STORE       !         set LINE# to FFFF so tha
               0
1890                *                                   R/W will REWIND the file
1891                *                                   and reread the line.
1892 E5A15 0000        CON(5)    =SEMI        ;         and return!
               0
```

```
1893                EJECT
1894            ***********************************************
1895            *
1896            * STRING  Create a string variable of a maximum
1897            *         length
1898            *
1899            * 10 STRING  HOWDY   (HOWDY has a max len of 10)
1900            *
1901            ***********************************************
1902
1903 E5A1A 7695          CON(5)  =!CLOSF
         E
1904 E5A1F 68    =!STRNG CON(2)  #86
1905 E5A21 3545          NIBASC  \STRIN\
         2594
         E4
1906 E5A2B 7C            CON(2)  \G\+#80
1907
1908 E5A2D 0000 =STRNG CON(5)  =DOCOL       :
         0
1909 E5A32 0405         CON(5)  =CREAT      CREATE
         E
1910 E5A37 49A5         CON(5)  =ROMC+      ROM;CODE
         E
1911            *                           with DOSTRING
1912 E5A3C 87A5         CON(5)  =M255       REPLACE MAXLEN WITH
         E
1913            *                           255 IF IT'S >255
1914 E5A41 0000         CON(5)  =DUP        maxlen maxlen
         0
1915 E5A46 0000         CON(5)  =HERE       maxlen maxlen here
         0
1916 E5A4B 0000         CON(5)  =C!        (store maxlen)
         0
1917 E5A50 0000         CON(5)  =ZERO       maxlen 0
         0
1918 E5A55 0000         CON(5)  =HERE       maxlen 0 here
         0
1919 E5A5A 0000         CON(5)  =TWO+       maxlen 0 here+2
         0
1920 E5A5F 0000         CON(5)  =C!        (store 0 as current length)
         0
1921 E5A64 0000         CON(5)  =TWO+       maxlen+2
         0
1922 E5A69 0000         CON(5)  =TWO*       2*(maxlen+2)=space to be
         0
1923            *                           allotted for chars and 2 len bytes
1924 E5A6E 3164         CON(5)  =ALLOT      NALLOT
         E
1925 E5A73 0000         CON(5)  =SEMI       ;
         0
1926
1927
1928            * REPLACE VALUE ON STACK WITH 255 IF IT'S >255
1929 E5A78 D7A5 =M255   CON(5)  =$M255
```



```
         E
1930 E5A7D 20    =$M255 P=      0
1931 E5A7F 143          A=DAT1  A
1932 E5A82 D2           C=0     A
1933 E5A84 31FF         LC(2)   255
1934 E5A88 8BA          ?A<=C   A           VALUE <256?
1935 E5A8B 40           GOYES   M255X       DON'T CHANGE
1936 E5A8D DA           A=C     A           REPLACE WITH 255
1937 E5A8F 141  M255X   DAT1=A  A
1938 E5A92 03           RTNCC
1939
1940 E5A94 0000 =ROMC+ CON(5)  =DOCOL       :
         0
1941 E5A99 0000         CON(5)  =ROM;C      ROM;CODE
         0
1942 E5A9E 0000         CON(5)  =DOSTR
         0
1943
1944 E5AA3 0000 =ROMC* CON(5)  =DOCOL       :
         0
1945 E5AA8 0000         CON(5)  =ROM;C      ROM;CODE
         0
1946 E5AAD 0000         CON(5)  =DOSTRA
         0
```

```
1947                    EJECT
1948                ***************************************************
1949                *
1950                * MAXLEN                    ( str  -- maxlen )
1951                * Maximum length of string. It is the first byte
1952                * at the pfa.
1953                *
1954                ***************************************************
1955
1956 E5AB2 F1A5          CON(5)  !STRNG
            E
1957 E5AB7 68    =!MXLEN CON(2)  #86
1958 E5AB9 D414          NIBASC  \MAXLE\
            85C4
            54
1959 E5AC3 EC            CON(2)  \N\+#80
1960
1961 E5AC5 0000 =MXLEN CON(5)  =DOCOL       :
            0
1962 E5ACA 0000          CON(5)  =DROP      DROP
            0
1963 E5ACF 0000          CON(5)  =LIT
            0
1964 E5AD4 4000          CON(5)  4          4
            0
1965 E5AD9 0000          CON(5)  =MINUS     -
            0
1966 E5ADE 0000          CON(5)  =CAT       C@
            0
1967 E5AE3 0000          CON(5)  =SEMI      ;
            0
1968
1969
1970                ***********************************************
1971                *
1972                * RIGHT$                    ( str size -- str )
1973                * Return a sub string of rightmost <size> chars.
1974                *
1975                * This is equivalent to doing <str> <len-size+1> SUB$
1976                * so that's how we implement it.
1977                ***********************************************
1978
1979 E5AE8 7BA5          CON(5)  =!MXLEN
            E
1980 E5AED 68    =!RITE$ CON(2)  #86
1981 E5AEF 2594          NIBASC  \RIGHT\
            7484
            45
1982 E5AF9 4A            CON(2)  \$\+#80
1983
1984 E5AFB 0000 =RITE$ CON(5)  =DOCOL       :
            0
1985 E5B00 0000          CON(5)  =OVER      addr len size len
            0
1986 E5B05 0000          CON(5)  =SWAP      addr len len size
```



```
            0
1987 E5B0A 0000          CON(5)  =MINUS     addr len (len-size)
            0
1988 E5B0F 0000          CON(5)  =ONE+      (len-size+1 = pos of 1st char)
            0
1989                * addr len 1st
1990 E5B14 0000          CON(5)  =OVER      addr len 1st len
            0
1991                * str 1st len
1992 E5B19 0000          CON(5)  =SUB$      SUB$
            0
1993 E5B1E 0000          CON(5)  =SEMI      ;
            0
1994
1995 E5B23          =REND4  END
```

=!?STK	Abs	938922	#E53AA	-	1366	1399					
=!ABRT"	Abs	939855	#E574F	-	1661	1743					
=!ALLOT	Abs	935429	#E4605	-	30	68					
=!BAS$	Abs	936021	#E4855	-	264	461					
=!BASCX	Abs	937005	#E4C2D	-	625	770					
=!BASF	Abs	935566	#E468E	-	103	263					
=!BASI	Abs	935478	#E4636	-	70	102					
=!CLOSF	Abs	940391	#E5967	-	1840	1903					
=!CMOV>	Abs	937830	#E4F66	-	1007	1031					
=!CREAT	Abs	938034	#E5032	-	1104	1197					
=!FENTR	Abs	936961	#E4C01	-	600	624					
=!FORGT	Abs	938487	#E51F7	-	1234	1365					
=!IOFIN	Abs	936727	#E4B17	-	506	528					
=!IOKIL	Abs	936785	#E4B51	-	529	553					
=!IOMAK	Abs	936607	#E4A9F	-	462	505					
=!LATE	Abs	939015	#E5407	-	1400	1417					
=!MXLEN	Abs	940727	#E5AB7	-	1957	1979					
=!NEGAT	Abs	937796	#E4F44	-	984	1006					
=!NMOV>	Abs	937866	#E4F8A	-	1032	1076					
=!NUMB	Abs	939113	#E5469	-	1451	1660					
=!OUTPT	Abs	937365	#E4D95	-	771	938					
=!REPET	Abs	938418	#E51B2	-	1198	1233					
=!RITE$	Abs	940781	#E5AED	-	1980						
=!SCFIB	Abs	937983	#E4FFF	-	1077	1103					
=!SHRNK	Abs	937673	#E4EC9	-	939	983					
=!SMUDG	Abs	940225	#E58C1	-	1769	1839					
=!SPACS	Abs	940146	#E5872	-	1744	1768					
=!STRNG	Abs	940575	#E5A1F	-	1904	1956					
=!TOGG	Abs	939059	#E5433	-	1418	1450					
=!UDIV	Abs	936842	#E4B8A	-	554	599					
$BASCX	Abs	937071	#E4C6F	-	645	644					
=$BASF	Abs	935585	#E46A1	-	108	106					
$BASI	Abs	935497	#E4649	-	75	73					
=$BSIZE	Abs	938018	#E5022	-	1088	1086					
=$BSUB	Abs	936160	#E48E0	-	295	294					
=$CLAP	Abs	936581	#E4A85	-	439	438					
=$CLOSF	Abs	940410	#E597A	-	1845	1843					
=$CMOV>	Abs	937849	#E4F79	-	1012	1011					
=$FENTR	Abs	936980	#E4C14	-	604	603					
$IOFIN	Abs	936746	#E4B2A	-	511	509					
$IOKIL	Abs	936804	#E4B64	-	533	532					
$IOMAK	Abs	936626	#E4AB2	-	467	465					
=$M255	Abs	940669	#E5A7D	-	1930	1929					
=$NEGAT	Abs	937815	#E4F57	-	989	987					
=$NMOV>	Abs	937885	#E4F9D	-	1037	1035					
$NMOVE	Ext			-	742						
$OUT	Abs	937426	#E4DD2	-	801	799					
=$SCFIB	Abs	938002	#E5012	-	1083	1080					
=$SHRNK	Abs	937692	#E4EDC	-	944	942					
=$UDIV	Abs	936861	#E4B9D	-	559	557					
2DROP	Ext			-	1139	1606	1609				
2DUP	Ext			-	287						
<>	Ext			-	1813						
>R	Ext			-	271	1204	1205	1273	1470	1534	1558	1729
					1868						

?COMP	Ext			-	1203						
?DUP	Ext			-	1348	1751					
=?PAIR	Abs	940269	#E58ED	-	1786						
=?STK	Abs	938936	#E53B8	-	1370						
AB0	Abs	940036	#E5804	-	1703	1692					
AB3	Abs	939991	#E57D7	-	1693	1732					
AB5	Abs	939924	#E5794	-	1678	1675					
AB6	Abs	939976	#E57C8	-	1690	1686					
ABER	Abs	940071	#E5827	-	1720	1718					
ABO	Abs	939894	#E5776	-	1672	1669					
ABORTW	Ext			-	631	786	1247	1263	1271	1629	1788
=ABRT"	Abs	939869	#E575D	-	1665						
ABRT"X	Ext			-	1129	1376	1387	1681			
=ABRTC	Abs	940046	#E580E	-	1715	1677					
ACTIV	Ext			-	635						
ADD	Ext			-	1385	1728	1810	1885			
ADHEAD	Ext			-	336						
AGAIN	Ext			-	1206						
=ALLOT	Abs	935443	#E4613	-	34	1159	1829	1924			
AT	Ext			-	37	630	1111	1134	1135	1141	1147	1152
					1173	1242	1243	1261	1276	1288	1304	1314
					1331	1347	1406	1407	1424	1458	1561	1568
					1573	1577	1673	1684	1866			
AVMEMS	Ext			-	147						
=B$SUB	Abs	936155	#E48DB	-	294	268					
BAD	Abs	937493	#E4E15	-	837						
BAD1	Abs	937503	#E4E1F	-	841	876					
=BASCX	Abs	937066	#E4C6A	-	644	633					
BASE	Ext			-	1560	1567	1572				
=BASF	Abs	935580	#E469C	-	106	1485					
=BASI	Abs	935492	#E4644	-	73						
BASIC	Ext			-	629						
=BASIC$	Abs	936035	#E4863	-	267						
BASIC?	Ext			-	1700						
=BASICX	Abs	937019	#E4C3B	-	628						
BL	Ext			-	1121	1239					
BRNCH	Ext			-	1315	1340	1512	1586	1610		
=BSIZE	Abs	938013	#E501D	-	1086						
BSU1	Abs	936204	#E490C	-	311	309					
BSX0	Abs	937120	#E4CA0	-	667	665					
BUF0	Abs	940460	#E59AC	-	1869	1871					
=BUFFR	Abs	940435	#E5993	-	1864						
C!	Ext			-	284	288	1156	1875	1916	1920	
CAT	Ext			-	1126	1150	1492	1502	1529	1559	1597	1725
					1806	1811	1826	1966			
CHK	Ext			-	944						
CLALL	Ext			-	1695						
=CLAP	Abs	936576	#E4A80	-	438	291					
CLEAN	Ext			-	1699						
CLOSEF	Ext			-	1848						
=CLOSF	Abs	940405	#E5975	-	1843						
CM3	Ext			-	1066						
=CMOV>	Abs	937844	#E4F74	-	1011						
CMOVE	Ext			-	276						
COLLAP	Ext			-	79	114	440				

COMMA	Ext			-	1120	1171					
COMPL	Ext			-	1676						
CONTX	Ext			-	1133	1275	1283	1295			
CORUPT	Ext			-	433						
COUNT	Ext			-	1180	1471	1484	1693			
CRE0	Abs	938093	#E506D	-	1121						
CRE01	Abs	938128	#E5090	-	1129						
CRE1	Abs	938135	#E5097	-	1132						
CREA10	Abs	938220	#E50EC	-	1149	1138	1143				
=CREAT	Abs	938048	#E5040	-	1108	1909					
CURR	Ext			-	1172	1241	1287	1405			
D+	Ext			-	1575						
DEFIN	Ext			-	1297	1698					
DIGIT	Ext			-	1562						
DNGAT	Ext			-	1649						
DOCOL	Ext			-	34	267	628	775	784	1108	1118	1179
					1202	1238	1370	1404	1422	1455	1462	1665
					1715	1748	1773	1786	1802	1825	1864	1908
					1940	1944	1961	1984				
DOSTR	Ext			-	1942							
DOSTRA	Ext			-	1946							
DOUSE	Ext			-	1083	1088						
DOVAR	Ext			-	1170							
DP	Ext			-	1355							
DPL	Ext			-	1482	1545	1576	1583	1633			
DROP	Ext			-	1249	1570	1639	1703	1733	1962		
DUP	Ext			-	270	278	1123	1124	1132	1144	1149	1154
					1160	1259	1265	1319	1326	1332	1469	1491
					1501	1527	1533	1557	1598	1612	1617	1667
					1716	1724	1803	1805	1867	1914		
EOLLEN	Ext			-	811							
EQUAL	Ext			-	1505	1532	1601	1615	1620	1624		
ERRTST	Ext			-	1670	1719						
EXCPAR	Ext			-	342							
EXEC	Ext			-	38	1112	1148	1459				
EXPR	Ext			-	396							
FAIL	Abs	936710	#E4B06	-	491	471	474	479	514	536		
FENCE	Ext			-	1260							
=FENTR	Abs	936975	#E4C0F	-	603							
FIND>	Ext			-	1136	1245						
FIVE-	Ext			-	1327	1339	1345	1354				
FL0	Abs	935743	#E473F	-	153	150						
FL1	Abs	935754	#E474A	-	163	168						
FLOT	Abs	935628	#E46CC	-	119	75	108	295				
FLOT0	Abs	935681	#E4701	-	133	128						
FLTDH	Ext			-	86							
=FLTOH	Abs	935533	#E466D	-	86	78						
=FORGT	Abs	938501	#E5205	-	1238							
FRGET1	Abs	938672	#E52B0	-	1286	1280						
FRGET2	Abs	938727	#E52E7	-	1303	1292						
FRGET3	Abs	938737	#E52F1	-	1308	1316						
FRGET4	Abs	938777	#E5319	-	1319	1313						
FRGET5	Abs	938802	#E5332	-	1331	1341						
FRGET6	Abs	938852	#E5364	-	1344	1338						
FRGETL	Abs	938792	#E5328	-	1326	1351						

FUNCD1	Ext			-	338	389					
=GCHAR	Abs	940291	#E5903	-	1802	1680					
GETADR	Abs	936561	#E4A71	-	430	406	409	412			
GETF#	Abs	935819	#E478B	-	193	76	109				
GETFP	Ext			-	81	115	318	441	481	491	516	537
					606	692	715	838	873	1849		
GSBSTK	Ext			-	372	402						
HERE	Ext			-	1164	1915	1918					
I/OALL	Ext			-	478	674						
I/ODAL	Ext			-	535							
=ICREAT	Abs	938078	#E505E	-	1118							
INBS	Ext			-	744							
=INUMB	Abs	939157	#E5495	-	1462							
IOFIN0	Abs	936775	#E4B47	-	517	540						
=IOFIND	Abs	936741	#E4B25	-	509							
IOFND0	Ext			-	513							
IOFSCR	Ext			-	473	672						
IOK0	Abs	936824	#E4B78	-	537							
=IOKILL	Abs	936799	#E4B5F	-	532							
=IOMAKE	Abs	936621	#E4AAD	-	465							
=IUNIQ	Abs	938355	#E5173	-	1179							
=LATE	Abs	939029	#E5415	-	1404	1119	1774					
LINE#	Ext			-	1888							
LINEP*	Ext			-	746							
LIT	Ext			-	35	279	1109	1145	1161	1166	1169	1281
					1293	1382	1456	1480	1503	1514	1530	1543
					1599	1613	1618	1622	1678	1775	1883	1886
					1963							
LTZ	Ext			-	1270							
=M255	Abs	940664	#E5A78	-	1929	1912						
M255X	Abs	940687	#E5A8F	-	1937	1935						
MAINST	Ext			-	971							
MAX	Ext			-	281	1750						
MB0	Abs	936705	#E4B01	-	489	493						
MEMERR	Ext			-	151							
MFERR	Ext			-	91	131	749					
MIN	Ext			-	1153							
MINUS	Ext			-	1787	1965	1987					
=MOVDP	Abs	940356	#E5944	-	1825	1687						
MOVED2	Ext			-	361							
MOVSET	Ext			-	1039							
MTHSTK	Ext			-	138	363	392					
MVMEM+	Ext			-	974							
=MXLEN	Abs	940741	#E5AC5	-	1961							
=NEGAT	Abs	937810	#E4F52	-	987							
=NIB7	Abs	937577	#E4E69	-	880	801						
NMOV0	Abs	937891	#E4FA3	-	1039	1015						
NMOV1	Abs	937919	#E4FBF	-	1049	1056						
NMOV2	Abs	937944	#E4FD8	-	1057	1050						
NMOV3	Abs	937947	#E4FDB	-	1058	1065						
NMOV4	Abs	937972	#E4FF4	-	1066	1059						
=NMOV>	Abs	937880	#E4F98	-	1035							
NUM00	Abs	939448	#E55B8	-	1543	1536						
NUM1	Abs	939845	#E5745	-	1650	1648						
=NUMB	Abs	939127	#E5477	-	1455							

NUMB00	Abs	939253	#E54F5	-	1488	1476					
NoCont	Ext			-	222	225					
OK	Abs	937555	#E4E53	-	866	831					
OK1	Abs	937511	#E4E27	-	845	868					
ONE	Ext			-	634	1582					
ONE+	Ext			-	1157	1578	1726	1807	1827	1988	
ONE-	Ext			-	1256	1509	1539				
OR	Ext			-	1625	1626					
OUTBYT	Ext			-	353						
=OUTP	Abs	937421	#E4DCD	-	799	776					
=OUTPUT	Abs	937379	#E4DA3	-	775						
OVER	Ext			-	1309	1344	1423	1809	1985	1990	
PAD	Ext			-	272	283	285	1381			
PBUF	Ext			-	1869						
PDOTQ	Ext			-	1185						
=PILERR	Abs	937399	#E4DB7	-	784	777					
POLL	Ext			-	628						
POPSTK	Ext			-	422						
POS	Ext			-	1472						
PREV	Ext			-	1880						
PSHSTK	Ext			-	374						
PSTOR	Ext			-	1584						
QUIT	Ext			-	1701						
QUOTC	Ext			-	1463						
R<RSTK	Ext			-	341						
R>	Ext			-	277	1207	1208	1353	1473	1605	1607	1608
					1636	1640	1723	1882				
R@	Ext			-	1274	1286	1308	1333	1596	1720	1874	1876
					1879							
=REND4	Abs	940835	#E5B23	-	1995							
=REPET	Abs	938432	#E51C0	-	1202							
REVPOP	Ext			-	337	426						
=RITE$	Abs	940795	#E5AFB	-	1984							
ROM;C	Ext			-	1941	1945						
=ROMC*	Abs	940707	#E5AA3	-	1944							
=ROMC+	Abs	940692	#E5A94	-	1940	1910						
ROT	Ext			-	1155	1468	1490	1571	1812			
RSTK<R	Ext			-	365							
Return	Abs	0	#00000	-	331	332						
S0	Ext			-	1372							
SAVEFP	Ext			-	120	439	467	511	533	604	655	880
					1847							
=SCFIB	Abs	937997	#E500D	-	1080							
SEMI	Ext			-	39	292	637	780	790	1113	1175	1189
					1211	1357	1391	1408	1428	1460	1486	1650
					1688	1704	1734	1759	1778	1793	1814	1830
					1892	1925	1967	1993				
SFLAG?	Ext			-	229	240						
SFLAGC	Ext			-	235	246						
SH0	Abs	937762	#E4F22	-	968	965						
SH09	Abs	937747	#E4F13	-	962	960						
=SHRINK	Abs	937687	#E4ED7	-	942							
=SMUDG	Abs	940239	#E58CF	-	1773							
SP!	Ext			-	1696							
SP@	Ext			-	1371	1380						

SPACE	Ext			-	1184	1756					
=SPACS	Abs	940160	#E5880	-	1748						
SPC0	Abs	940215	#E58B7	-	1759	1753					
SPC1	Abs	940200	#E58A8	-	1756	1758					
SS1	Abs	935895	#E47D7	-	221	218					
SS2	Abs	935909	#E47E5	-	226	223					
SS3	Abs	935952	#E4810	-	237	231					
SS4	Abs	935995	#E483B	-	248	242					
STATE	Ext			-	1672	1683					
STKCHR	Ext			-	335						
STKLFT	Ext			-	110	605					
STMTR0	Ext			-	813	845					
STMTR1	Ext			-	807						
STORE	Ext			-	636	1174	1284	1296	1321	1346	1356	1427
					1483	1546	1634	1873	1878	1881	1889	
=STRNG	Abs	940589	#E5A2D	-	1908							
SUB$	Ext			-	1992							
SUCC	Abs	936698	#E4AFA	-	486							
=SVSTAT	Abs	935879	#E47C7	-	215							
SWAP	Ext			-	275	290	1373	1426	1474	1493	1508	1510
					1538	1540	1552	1566	1616	1621	1690	1721
					1986							
THEN	Ext			-	1210							
=TOGG	Abs	939073	#E5441	-	1422	1163	1168	1777				
TRAVS	Ext			-	1258							
TWO*	Ext			-	1158	1727	1808	1828	1922			
TWO+	Ext			-	273	274	286	289	1125	1500	1528	1537
					1556	1877	1919	1921				
TWO-	Ext			-	1165	1209	1254	1518				
TYPE	Ext			-	1183	1694						
U*	Ext			-	1569	1574						
U<	Ext			-	1262	1277	1289	1310	1334	1374	1386	
=UDIV	Abs	936856	#E4B98	-	557							
UDIV0	Abs	936899	#E4BC3	-	572	575						
UDIV1	Abs	936905	#E4BC9	-	574	569						
UDIV2	Abs	936913	#E4BD1	-	577	584						
UDIV3	Abs	936930	#E4BE2	-	583	576	580					
UDIV4	Abs	936936	#E4BE8	-	585	571						
USE	Ext			-	1865	1872						
=VAL0$	Abs	936251	#E493B	-	332	296						
VAL00	Ext			-	193							
VAL01	Abs	936327	#E4987	-	350							
VALERR	Abs	936247	#E4937	-	329	347	349	351				
VOC-L	Ext			-	1303	1320						
=ValSub	Abs	10	#0000A	-	326							
WARN	Ext			-	1140							
WIDTH	Ext			-	1151							
WORD	Ext			-	1122	1240	1804					
WR1	Ext			-	966							
XDO	Ext			-	1554	1755						
XLOOP	Ext			-	1637	1757						
XOR	Ext			-	1425							
XXHEAD	Ext			-	333							
ZBRNH	Ext			-	1137	1142	1279	1291	1311	1337	1350	1475
					1506	1516	1535	1564	1580	1603	1647	1668

```
                                  1674  1685  1691  1717  1731  1752  1870
ZEQ      Ext                   -  785  1127  1246  1335  1349  1627  1730
ZERO     Ext                   -  1255 1488  1489  1511  1553  1632  1749  1754
                                  1917
adj      Ext                   -  975
eDATTY   Ext                   -  130
eIVerr   Abs  935665 #E46F1   -  129   214
eIVgo    Abs  935875 #E47C3   -  214   202   329
eMEM     Ext                   -  748
eOVFLW   Ext                   -  90
flPRGM   Ext                   -  228   234
flSUSP   Ext                   -  238   245
nom      Abs  937156 #E4CC4   -  677   673
nomem    Abs  937349 #E4D85   -  748   677
numy     Abs  939483 #E55DB   -  1556  1638
numy0    Abs  939628 #E566C   -  1596  1565
numy1    Abs  939618 #E5662   -  1586  1581
numy2    Abs  939805 #E571D   -  1635  1587
numy22   Abs  939825 #E5731   -  1640  1611
numy3    Abs  939703 #E56B7   -  1612  1604
numz     Abs  939283 #E5513   -  1500  1517
numz1    Abs  939353 #E5559   -  1514  1507
numz2    Abs  939363 #E5563   -  1516  1513
pPRTCL   Ext                   -  829
tEOL     Ext                   -  345   352
yesmem   Abs  937160 #E4CC8   -  681   675
```


Input Parameters

Source file name is MR&FT4

Listing file name is MR/FT4

Object file name is MR%FT4::65

```
                                  111111
                        0123456789012345
```

Initial flag settings are

Errors

None

Saturn Assembler News


```
 1                      TITLE  FT5:_7_&_8_CHAR_WORDS
 2                      RDSYMB MR%GIO
 3                *     RDSYMB MR%FT4
 4                *     ABS    REND4
 5
 6 E5C00                ABS    #E5C00
 7
 8                ******************************************
 9                *
10                * MR&FT5     <840608.1403>
11                *
12                *
13                *   START OF 7 CHAR WORDS
14                * ADD ADDITIONAL WORDS AT THIS END
15                *
16                ******************************************
17
18                ******************************************
19                *
20                * LISTING                    ( -- str)
21                *
22                * This is a string variable, the only one
23                * built into the system.
24                *
25                ******************************************
26 E5C00 0000          CON(5)  0            NO MORE 7'S
         0
27 E5C05 78    =!LSTNG CON(2)  #87
28 E5C07 C494          NIBASC  \LISTIN\
         3545
         94E4
29 E5C13 7C            CON(2)  \G\+#80
30 E5C15 A1C5  =LISTNG CON(5)  =$LSTNG
         E
31 E5C1A 20    =$LSTNG P=      0
32 E5C1C 34D4          LC(5)   =oLSTNG     GET ADDR OF VARIABLE
         CF2
33 E5C23 E6            C=C+1   A
34 E5C25 E6            C=C+1   A            POINT TO CURLEN
35 E5C27 137           CD1EX
36 E5C2A D0            A=0     A
37 E5C2C 14B           A=DAT1  B            GET CUR LEN
38 E5C2F 137           CD1EX
39 E5C32 E6            C=C+1   A
40 E5C34 E6            C=C+1   A            POINT TO BEGINNING OF STR
41 E5C36 1C4           D1=D1-  5
42 E5C39 145           DAT1=C  A            PUT ADDR ON STK
43 E5C3C 1C4           D1=D1-  5
44 E5C3F 141           DAT1=A  A            PUT LEN ON STK
45 E5C42 03            RTNCC
```

```
                        EJECT
                *****************************************************
                *
                * PRIMARY ( addr --- ) Returns the addr of a variable
                *       which contains the HPIL PRIMARY address (0, 31)
                *       of the device from which data will be ENTERed
                *       and OUTPUT.
                *
                *       It defaults to 1.  No checking is done to
                *       make sure that the user does not screw things
                *       up.
                *
                *****************************************************

E5C44 50C5E     =!LSTNG CON(5)  =!LSTNG
E5C49 78        =!PRIME CON(2)  #87
E5C4B 0525      94D41425        NIBASC  \PRIMAR\
E5C57 9D        =PRIME  CON(2)  \Y\+#80
E5C59 E5C5E     =PRIME  CON(5)  =$PRIME

E5C5E 8E0000    =$PRIME GOSUBL  =DOUSE
E5C64 1BBF2     =oPRIME CON(5)  (=oPRIME)
```



```
                        EJECT
```

```
                        EJECT
                *********************************************
                *
                * EXECUTE  (addr --- )  Execute the dictionary
                *        entry whose cfa addr is on the stack.
                *
                *********************************************

E5C69 94C5E             CON(5)  !PRIME
E5C6E 78        =!EXEC  CON(2) #87
E5C70 5485              NIBASC \EXECUT\
      5434
      5545
E5C7C 5C                CON(2) \E\+#80
E5C7E 38C5E =EXEC       CON(5)  $EXEC

E5C83 143       $EXEC   A=DAT1 A                @DATA STK--> WA
E5C86 174               D1=D1+ 5                JUMP INTO NEXT AT RUN
E5C89 8C000             GOLONG =RUN
```



```
                        EJECT
                *********************************************
                *
                * RADIANS       SET MACHINE RADIAN MODE
                *
                *********************************************
E5C8F E6C5E             CON(5)  !EXEC
E5C94 78        =!RAD   CON(2)  #87
E5C96 2514              NIBASC  \RADIAN\
      4494
      14E4
E5CA2 3D                CON(2)  \S\+#80
E5CA4 9AC5E =RAD        CON(5)  =$RAD
E5CA9 8E000 =$RAD       GOSUBL  =SAVEFP
E5CAF 3100              LC(2)   =flRAD
E5CB3 8F000             GOSBVL  =SFLAGS
E5CBA 8C000             GOLONG  =GETFP

                *********************************************
                *
                * DEGREES       SET MACHINE DEGREE MODE
                *
                *********************************************
E5CC0 49C5E             CON(5)  =!RAD
E5CC5 78        =!DEGR  CON(2)  #87
E5CC7 4454              NIBASC  \DEGREE\
      7425
      5454
E5CD3 3D                CON(2)  \S\+#80
E5CD5 ADC5E =DEGR       CON(5)  =$DEGR
E5CDA 8E000 =$DEGR      GOSUBL  =SAVEFP
E5CE0 3100              LC(2)   =flRAD
E5CE4 8F000             GOSBVL  =SFLAGC     CLEAR RADIANS FLAG
E5CEB 8C000             GOLONG  =GETFP
```

```
116                     EJECT
117                 ********************************************************
118                 *
119                 * DNEGATE (d1 ---  -d1)  leave 2's comp of double number
120                 *
121                 ********************************************************
122
123 E5CF1 5CC5             CON(5)  =!DEGR
        E
124 E5CF6 78        =!DNGAT CON(2)  #87             DNEGATE
125 E5CF8 44E4           NIBASC  \DNEGAT\
        5474
        1445
126 E5D04 5C             CON(2)  \E\+#80
127 E5D06 B0D5 =DNGAT    CON(5)  =$DNGAT
        E
128
129 E5D0B 1C4   =$DNGAT  D1=D1-  5
130 E5D0E 15F9           C=DAT1  10             MSDs to C9:5
131 E5D12 179            D1=D1+  10
132 E5D15 147            C=DAT1  A              LSDs to C4:0
133 E5D18 29             P=      9
134 E5D1A B9A            C=-C    WP             2'S COMP
135 E5D1D 145            DAT1=C  A              LSDs back to the stack
136 E5D20 1C9            D1=D1-  10
137 E5D23 15D9           DAT1=C  10             MSDs back to the stack
138 E5D27 174            D1=D1+  5
139 E5D2A 03             RTNCC
```



```
140                     EJECT
141                 ****************************************
142                 *
143                 * RECSIZE  ( --- addr)  addr of variable
144                 *       which contains the max size of
145                 *       disc buffer record
146                 *
147                 ****************************************
148
149 E5D2C 13D5 =RSIZE  CON(5)  =$RSIZE
        E
150 E5D31 8E00 =$RSIZE GOSUBL  =DOUSE
        00
151 E5D37 93BF         CON(5)   (=oRSIZE)
        2
152
153
154                 ****************************************
155                 *
156                 * CONTEXT  ( --- addr)  addr of variable
157                 *       which contains the addr of the
158                 *       current vocabulary for searches
159                 *
160                 ****************************************
161
162 E5D3C 6FC5             CON(5)  !DNGAT
        E
163 E5D41 78        =!CONTX CON(2)  #87             CONTEXT
164 E5D43 34F4           NIBASC  \CONTEX\
        E445
        5485
165 E5D4F 4D             CON(2)  \T\+#80
166 E5D51 65D5 =CONTX    CON(5)  =$CONTX
        E
167
168 E5D56 8E00 =$CONTX GOSUBL  =DOUSE
        00
169 E5D5C 66BF         CON(5)   (=oCONTX)
        2
```

```
                        EJECT
                *****************************************
                *
                * CURRENT  ( --- addr)  addr of variable
                *       which contains the addr of the
                *       vocabulary to which new entries
                *       are added
                *
                *****************************************

E5D61 14D5E             CON(5)  !CONTX
E5D66 78        =!CURR  CON(2)  #87             CURRENT
E5D68 3455              NIBASC  \CURREN\
      2525
      54E4
E5D74 4D                CON(2)  \T\+#80
E5D76 B7D5E     =CURR   CON(5)  =$CURR

E5D7B 8E00      =$CURR  GOSUBL  =DOUSE
      00
E5D81 B6BF2             CON(5)   (=oCURR)
```



```
                        EJECT
                *****************************************************
                *
                * 0BRANCH   ( flag --- )  If the flag is false, pick up
                *           offset following 0BRANCH and add it to I
                *           otherwise increment I to skip past the increment.
                *
                *
                *                       0BRANCH
                *           I------>    <offset>
                *                       DUP
                *
                *************************************************

E5D86 B8D5E     =ZBRNH  CON(5) =$ZBRNH

E5D8B 143       =$ZBRNH A=DAT1  A               A=FLAG
E5D8E 174               D1=D1+  5               RETURN NOTHING ON DATA STK
E5D91 8AC               ?A#0    A               TRUE?
E5D94 D1                GOYES   0B0             YES
E5D96 570               GONC    $BRNCH

E5D99 E9D5E     =BRNCH  CON(5) =$BRNCH

E5D9E 146       =$BRNCH C=DAT0  A               READ OFFSET TO I
E5DA1 132               ADOEX                   A=CURRENT I
E5DA4 C2                C=C+A   A               NEW I
E5DA6 134               D0=C                    STORE NEW I
E5DA9 8E00              GOSUBL  =DOATN          CHECK FOR ATTN KEY
      00
E5DAF 03                RTNCC

E5DB1 164       0B0     D0=D0+  5               POINT BEYOND OFFSET
E5DB4 8E00              GOSUBL  =DOATN          CHECK FOR ATTN HIT, DO ABORT
      00
E5DBA 03                RTNCC

                ********************************************
                *
                * DECIMAL    ( --- )  Set BASE to decimal mode
                *
                ********************************************

E5DBC 66D5E             CON(5)  =!CURR
E5DC1 78        =!DEC   CON(2)  #87             DECIMAL
E5DC3 4454              NIBASC  \DECIMA\
      3494
      D414
E5DCF CC                CON(2)  \L\+#80

E5DD1 0000      =DEC    CON(5)  =DOCOL          :
      0
```

```
235 E5DD6 0000          CON(5)   =LIT
          0
236 E5DDB A000          CON(5)  10            10
          0
237 E5DE0 0000          CON(5)   =BASE        BASE
          0
238 E5DE5 0000          CON(5)   =STORE       !
          0
239 E5DEA 0000          CON(5)   =SEMI        ;
          0
```



```
240             EJECT
241             ************************************************
242             *
243             * CONVERT  (d1 addr1 --- d2 addr2) Converts the text beginning
244             *          at addr1+2 (NIBS) to a binary value with regard to
245             *          BASE.  New value is accumulated into d2; addr2 is 1st
246             *          non-convertible char.
247             *
248             ************************************************************************
249
250 E5DEF 1CD5          CON(5)  =!DEC
          E
251 E5DF4 78   =!CONVT CON(2)  #87             CONVERT
252 E5DF6 34F4        NIBASC  \CONVER\
          E465
          5425
253 E5E02 4D          CON(2)  \T\+#80
254
255 E5E04 0000 =CONVT  CON(5)  =DOCOL          :
          0
256 E5E09 0000 CONV1   CON(5)   =TWO+          2+  INCR BY 2 TO GET TO NEXT BYT
          0
257 E5E0E 0000         CON(5)   =DUP        d1 addr+2 addr+2
          0
258 E5E13 0000         CON(5)   =>R         d1 addr+2
          0
259 E5E18 0000         CON(5)   =CAT        d1 char
          0
260 E5E1D 0000         CON(5)   =BASE           BASE
          0
261 E5E22 0000         CON(5)   =AT         d1 ch base
          0
262 E5E27 0000         CON(5)   =DIGIT      return true if digit , plus binary va
          0
263 E5E2C 68D5         CON(5)   =ZBRNH          WHILE
          E
264 E5E31 6400         CON(5)  (CONV3)-*
          0
265 E5E36 0000         CON(5)   =SWAP       d1a n d1b
          0
266 E5E3B 0000         CON(5)   =BASE           BASE
          0
267 E5E40 0000         CON(5)   =AT         d1a n d1b base
          0
268 E5E45 0000         CON(5)   =U*         d1a n base*d1b
          0
269 E5E4A 0000         CON(5)   =DROP           drop top of double-num result
          0
270 E5E4F 0000         CON(5)   =ROT        n d1b d1a
          0
271 E5E54 0000         CON(5)   =BASE       n d1b d1a base
          0
272 E5E59 0000         CON(5)   =AT         n d1a d1b [base]
          0
273 E5E5E 0000         CON(5)   =U*         n d1a <base*d1b>
```

```
            0
274 E5E63 0000          CON(5)   =D+        d2
            0
275 E5E68 0000          CON(5)   =R>        d2 addr+2(char-ptr)
            0
276             *                           REPEAT
277 E5E6D 99D5          CON(5)   =BRNCH
            E
278 E5E72 79FF          CON(5)   (CONV1)-*
            F
279 E5E77 0000 CONV3    CON(5)   =R>          R>
            0
280 E5E7C 0000          CON(5)   =SEMI
            0
281
```



```
282             EJECT
283           ***************************************************************
284           *
285           * ENCLOSE (addr1 c --- addr1 n1 n2 n3) From addr1 and an
286           *         ASCII delimiting char, c, is determined the NIBBLE
287           *         offset to the first non-delimiter char, n1, the offset
288           *         to the 1st delimiter after the text, n2, and the offset
289           *         to the 1st char not included, n3. An ASCII null is an
290           *         unconditional delimiter.
291           *
292           ***************************************************************
293
294 E5E81 4FD5          CON(5)   =!CONVT
            E
295 E5E86 78    =!ENCL  CON(2)   #87            ENCLOSE
296 E5E88 54E4          NIBASC   \ENCLOS\
            34C4
            F435
297 E5E94 5C            CON(2)   \E\+#80
298 E5E96 B9E5 =ENCL    CON(5)   =$ENCL
            E
299
300 E5E9B 143  =$ENCL   A=DAT1   A              A=CHAR
301 E5E9E D9            C=B      A
302 E5EA0 109           R1=C                   SAVE RTN STK IN R1
303 E5EA3 D1            B=0      A
304 E5EA5 174           D1=D1+   5             PT TO ADDR
305 E5EA8 147           C=DAT1   A             C=ADDR
306 E5EAB 1C4           D1=D1-   5             PT TO CHAR AGAIN
307 E5EAE 136           CD0EX                  D0=ADDR
308 E5EB1 10A           R2=C                   SAVE I IN R2
309 E5EB4 181           D0=D0-   2             SET MEMORY PTR FOR LOOP
310 E5EB7 CD            B=B-1    A
311 E5EB9 CD            B=B-1    A             SET COUNTER FOR LOOP
312 E5EBB 161  ENCL1    D0=D0+   2             PT TO NEXT CHAR
313 E5EBE 14E           C=DAT0   B             C(B)=CHAR
314 E5EC1 E5            B=B+1    A
315 E5EC3 E5            B=B+1    A             INCREMENT COUNTER
316 E5EC5 962           ?C=A     B             =DELIMITER?
317 E5EC8 3F            GOYES    ENCL1         YES, GET ANOTHER
318 E5ECA DD            BCEX     A             COUNTER TO C
319 E5ECC 145           DAT1=C   A             PUSH N1
320 E5ECF 1C4           D1=D1-   5             READY FOR N2
321 E5ED2 96D           ?B#0     B             NON-NULL CHAR?
322 E5ED5 01            GOYES    ENCL2         YES
323             *                              WAS NULL
324 E5ED7 E6            C=C+1    A             INCREMENT COUNTER
325 E5ED9 E6            C=C+1    A             BY 1 BYTE
326 E5EDB 145           DAT1=C   A             PUSH N2
327 E5EDE CE            C=C-1    A
328 E5EE0 CE            C=C-1    A             DECREMENT COUNTER
329 E5EE2 592           GONC     ENCL4         FINISH HERE
330 E5EE5 DD   ENCL2    BCEX     A             COUNTER TO B
331 E5EE7 161  ENCL20   D0=D0+   2             PT TO NEXT CHAR
332 E5EEA E5            B=B+1    A             INCR COUNTER
```

```
 333 E5EEC E5           B=B+1   A           INCR COUNTER
 334 E5EEE 14E          C=DAT0  B           C(B)=CHAR
 335 E5EF1 962          ?C=A    B           =DELIMITER?
 336 E5EF4 F0           GOYES   ENCL3       YES
 337 E5EF6 96E          ?C#0    B           NON-NULL CHAR?
 338 E5EF9 EE           GOYES   ENCL20      YES, GET ANOTHER
 339 E5EFB DD           BCEX    A           COUNTER TO C
 340 E5EFD 145          DAT1=C  A           PUSH N2
 341 E5F00 5B0          GONC    ENCL4       FINISH HERE
 342 E5F03 D9   ENCL3   C=B     A
 343 E5F05 145          DAT1=C  A           PUSH N2
 344 E5F08 E6           C=C+1   A
 345 E5F0A E6           C=C+1   A           INCR CNTR 1 BYTE
 346 E5F0C 1C4  ENCL4   D1=D1-  5
 347 E5F0F 145          DAT1=C  A           PUSH N3
 348 E5F12 11A          C=R2
 349 E5F15 136          CD0EX               RESTORE I TO D0
 350 E5F18 119          C=R1
 351 E5F1B D5           B=C     A           RESTORE RTN STK
 352 E5F1D 03           RTNCC
```



```
 353                    EJECT
 354                 ******************************************************
 355                 *
 356                 * 'STREAM ( --- addr) Returns addr of the next char
 357                 *         in the input stream.
 358                 *
 359                 *  We assume that >IN is being incremented by 2 for
 360                 *  nibbles as opposed to 1 for bytes!
 361                 *
 362                 ******************************************************
 363 E5F1F 68E5             CON(5)  =!ENCL
         E
 364 E5F24 78     =!STRM    CON(2)  #87             'STREAM
 365 E5F26 7235            NIBASC  \'STREA\
         4525
         5414
 366 E5F32 DC              CON(2)  \M\+#80
 367
 368 E5F34 0000   =STRM    CON(5)  =DOCOL          :
         0
 369 E5F39 0000            CON(5)   =BLK           BLK   if contents of BLK are non-
         0
 370 E5F3E 0000            CON(5)   =AT            @     then they represent the lin
         0
 371 E5F43 0000            CON(5)   =?DUP          ?DUP  in a file which we are inte
         0
 372 E5F48 68D5            CON(5)   =ZBRNH         IF    if this is the case BLK#0 t
         E
 373 E5F4D 4100            CON(5)  (ST0)-*               use BLOCK to return addr of
         0
 374 E5F52 0000            CON(5)   =BLOCK         BLOCK to be intrepreted
         0
 375 E5F57 99D5            CON(5)   =BRNCH         ELSE  otherwise we are interpreti
         E
 376 E5F5C A000            CON(5)  (ST01)-*              from the TIB
         0
 377 E5F61 0000 ST0        CON(5)   =TIB           TIB   so get its addr
         0
 378 E5F66 0000 ST01       CON(5)   =IN            THEN  get contents of >IN (which
         0
 379 E5F6B 0000            CON(5)   =AT            @     is the offset into the curr
         0
 380 E5F70 0000            CON(5)   =ADD           +     line) and add to base addr
         0
 381 E5F75 0000            CON(5)   =SEMI          ;
         0
```

```
                        EJECT
                *************************************************************
                *
                * LITERAL (n --- ) If compiling, then compile the stack
                *       value as a 20 bit literal which when later executed
                *       will leave "n" on the stack
                *
                *        IMMEDIATE
                *
                *************************************************************
E5F7A 42F5      =!STRM  CON(5)  =!STRM
      E
E5F7F 7C        =!LITR  CON(2)  #C7             LITERAL
E5F81 C494              NIBASC  \LITERA\
      4554
      2514
E5F8D CC                CON(2)  \L\+#80

E5F8F 0000      =LITR   CON(5)  =DOCOL          :
      0
E5F94 0000              CON(5)  =STATE          STATE   if state is 0
      0
E5F99 0000              CON(5)  =AT             @       do nothing
      0
E5F9E 68D5              CON(5)  =ZBRNH          IF      otherwise
      E
E5FA3 4100              CON(5)  (LITR0)-*
      0
E5FA8 1DF5              CON(5)   =COMPL         COMPILE literal handler
      E
E5FAD 0000              CON(5)   =LIT           LIT     runtime code
      0
E5FB2 0000              CON(5)   =COMMA         ,       into dictionary
      0
E5FB7 0000      LITR0   CON(5)   =SEMI          ;       as well as value
      0
```



```
                        EJECT
                *****************************************
                *
                * COMPILE
                *
                * used in form:   : <name> . . . COMPILE <namex> ;
                *
                * When <name> is executed the compilation addr compiled
                * for <namex> is compiled and NOT executed. <name> is
                * typically immediate and <namex> is typically not
                * immediate.
                *
                *****************************************

E5FBC F7F5              CON(5)  !LITR
      E
E5FC1 78        =!COMPL CON(2)  #87             COMPILE
E5FC3 34F4              NIBASC  \COMPIL\
      D405
      94C4
E5FCF 5C                CON(2)  \E\+#80

E5FD1 0000      =COMPL  CON(5)  =DOCOL
      0
E5FD6 0000              CON(5)  =?COMP  error if not compiling
      0
E5FDB 0000              CON(5)  =R>     get return addr; copy it
      0
E5FE0 0000              CON(5)  =DUP    add 5 to skip beyond value to
      0
E5FE5 0000              CON(5)  =FIVE+  compile and push it back to
      0
E5FEA 0000              CON(5)  =>R     return stack; use unchanged
      0
E5FEF 0000              CON(5)  =AT     addr, get contents and store in
      0
E5FF4 0000              CON(5)  =COMMA  dictionary.
      0
E5FF9 0000              CON(5)  =SEMI
      0
```

```
                        EJECT
                *************************************************
                *
                * ?NEGATE   (n1 n2 -- n3) if n2 < 0 then NEGATE n1
                *
                * MADE HEADERLESS 10/10/83
                *
                *************************************************

E5FFE 00000 =?NEG   CON(5)  =DOCOL          :
E6003 00000         CON(5)   =LTZ           0<
E6008 68D5E         CON(5)   =ZBRNH         IF
E600D A0000         CON(5)  (NG0)-*
E6012 00000         CON(5)   =NEGAT         NEGATE
E6017 00000 NG0     CON(5)   =SEMI          THEN ;



                ***********************************
                *
                * <;CODE> FOR RAM DEFINING WORDS
                *  (SEE P. 54 IN ALL ABOUT FORTH)
                *
                ***********************************

E601C 00000 =<;COD  CON(5)  =DOCOL          :
E6021 00000         CON(5)   =R>            R>
E6026 00000         CON(5)   =LATE          LATEST
E602B 00000         CON(5)   =PFA           PFA
E6030 00000         CON(5)   =LIT
E6035 50000         CON(5)   5              5
E603A 00000         CON(5)   =MINUS         -    (RAM CFA ADDR)
E603F 00000         CON(5)   =STORE         !
E6044 00000         CON(5)   =SEMI          ;
```



```
                        EJECT
                *****************************************
                *
                * ENDCASE  (FROM CASE OF ENDOF ENDCASE)
                *
                *****************************************

E6049 1CF5E         CON(5)  =!COMPL
E604E 7C    =!ENDCS CON(2)  #C7
E6050 54E44434 1435 NIBASC  \ENDCAS\
E605C 5C            CON(2)  \E\+#80

E605E 00000 =ENDCS CON(5) =DOCOL        :
E6063 00000         CON(5)  =?COMP      ensure compile mode
E6068 00000         CON(5)  =DUP
E606D 00000        CON(5)   =LIT
E6072 70000        CON(5)  7            7 LEFT BY ENDOF

                * THERE MUST HAVE BEEN AT LEAST ONE OF..ENDOF
E6077 00000         CON(5)  =<>         not equal
E607C 00000         CON(5)  =ABRT"X     abort if not 7
E6081 C3            CON(2)  =eCASE
E6083 1DF5E        CON(5)   =COMPL      COMPILE a DROP
E6088 00000        CON(5)   =DROP       to drop the unmatched index value if
                *                       gets that far at runtime
                *                       BEGIN
E608D 00000 ENDC1  CON(5)   =SP@        SP@
E6092 00000        CON(5)   =CSP        CSP
E6097 00000        CON(5)   =AT         @
E609C 00000        CON(5)   =EQUAL      =
E60A1 68D5E        CON(5)   =ZBRNH      IF THERE WAS A 7 BEFORE THE CASE
E60A6 A0000        CON(5)  (ENDC2)-*
E60AB 00000         CON(5)  =SEMI           THEN JUST QUIT
E60B0 00000 ENDC2   CON(5)  =DUP
E60B5 00000         CON(5)  =LIT
```

```
502 E60BA 7000          CON(5)  7
          0
503 E60BF 0000          CON(5)  =EQUAL          MATCH ANOTHER 7 FROM ENDOF?
          0
504 E60C4 68D5          CON(5)  =ZBRNH        WHILE
          E
505 E60C9 E100          CON(5)  (ENDC3)-*
          0
506 E60CE 0000          CON(5)  =DROP           DROP THE 7
          0
507 E60D3 0000        CON(5)    =TWO        2
          0
508             *     CON(5)    =[CMP]      [COMPILE]
509 E60D8 0000        CON(5)    =THEN       FILL IN HERE AT THE ENDOF ADDR
          0
510 E60DD 99D5        CON(5)    =BRNCH      REPEAT
          E
511 E60E2 BAFF        CON(5)  (ENDC1)-*
          F
512 E60E7 0000 ENDC3  CON(5)    =SEMI       ;
          0
```



```
513                 EJECT
514           ****************************************
515           *
516           * Renamed CREATEF (from FCREATE) on
517           * 12/18/83
518           *
519           * CREATEF       ( str size -- addr )
520           *               ( str size -- f    )
521           * Create a file of type text, with
522           * given name and size.
523           *
524           ****************************************
525 E60EC E406          CON(5)  =!ENDCS
          E
526 E60F1 78    =!FCRT  CON(2)  #87
527 E60F3 3425          NIBASC  \CREATE\
          5414
          4554
528 E60FF 6C            CON(2)  \F\+#80
529 E6101 0000 =FCREAT CON(5)  =DOCOL
          0
530 E6106 0000          CON(5)  =ROT
          0
531 E610B 0000          CON(5)  =ROT            SIZE STR
          0
532 E6110 0000          CON(5)  =2DUP           SIZE STR STR
          0
533 E6115 0000          CON(5)  =FFIND           SIZE STR FLAG
          0
534 E611A 68D5          CON(5)  =ZBRNH          OK IF NOT FOUND
          E
535 E611F 9100          CON(5)  (FCR02)-*
          0
536 E6124 0000 FCFAIL   CON(5)  =2DROP
          0
537 E6129 0000 FCR01    CON(5)  =DROP
          0
538 E612E 0000          CON(5)  =ZERO       RETURN FALSE FLAG
          0
539 E6133 0000          CON(5)  =SEMI
          0
540
541 E6138 B126 FCR02    CON(5)  =FSTX           SIZE FLAG
          E
542 E613D 68D5          CON(5)  =ZBRNH          IF OK
          E
543 E6142 7EFF          CON(5)  (FCR01)-*
          F
544 E6147 6516          CON(5)  =FNAME          SIZE FNAME
          E
545 E614C B826          CON(5)  =CRFIL          ADDR  OR FALSE FLAG
          E
546 E6151 0000          CON(5)  =SEMI
          0
547
```

```
548
```



```
549                     EJECT
550                 ********************************************
551                 * UTILITIES FOR FCREATE
552                 ********************************************
553 E6156 B516 =FNAME  CON(5)  =$FNAM
          E
554 E615B 8E00 =$FNAM  GOSUBL  =DOUSE
          00
555 E6161 01CF        CON(5)  =oFNAME
          2
556
557                 ****************************************
558                 *
559                 * SETNAM        ( str -- )
560                 * Set up a string in the format required
561                 * by FSPECx, that is, with quotes around
562                 * it and a terminator byte, at STMTR0.
563                 *
564                 * ONLY USES 1st 8 CHARS OF NAME!
565                 *
566                 ****************************************
567 E6166 0000 =SETNAM CON(5)  =DOCOL
          0
568 E616B 0000        CON(5)  =ABS            ABS VALUE OF LENGTH OF STRING
          0
569 E6170 0000        CON(5)  =LIT            MAKE SURE THAT
          0
570 E6175 8000        CON(5)  8               FILE NAME IS NOT
          0
571 E617A 0000        CON(5)  =MIN            LONGER THAN 8 CHARS!
          0
572 E617F 0000        CON(5)  =DUP            STR LEN
          0
573 E6184 0000        CON(5)  =TWO*           STR LEN*2
          0
574 E6189 0000        CON(5)  =LIT
          0
575 E618E 0000        CON(5)  =STMTR0         STR LEN*2 SR0
          0
576 E6193 0000        CON(5)  =DUP            STR LEN*2 SR0 SR0
          0
577 E6198 0000        CON(5)  =ROT            STR SR0 SR0 LEN*2
          0
578 E619D 0000        CON(5)  =ADD            STR SR0 END-OF-STRING
          0
579 E61A2 0000        CON(5)  =>R             STR SR0  :: EOS
          0
580 E61A7 0000        CON(5)  =LIT            STR SR0  :: EOS
          0
581 E61AC 2200        CON(5)  34              STR SR0 "
          0
582 E61B1 0000        CON(5)  =OVER           STR SR0 " SR0
          0
583               * STORE BEGINNING QUOTE
584 E61B6 0000        CON(5)  =C!             STR SR0
```

```
          0
585 E61BB 0000        CON(5)  =TWO+           STR SR0+2
          0
586 E61C0 0000        CON(5)  =SMOVE
          0
587 E61C5 0000        CON(5)  =R@             EOS         :: EOS
          0
588 E61CA 0000        CON(5)  =TWO+           (ACCOUNT FOR ")
          0
589 E61CF 0000        CON(5)  =LIT
          0
590 E61D4 2200        CON(5)  34              EOS "
          0
591 E61D9 0000        CON(5)  =SWAP           " EOS
          0
592 E61DE 0000        CON(5)  =C!             STORED END QUOTE
          0
593 E61E3 0000        CON(5)  =R>             EOS
          0
594 E61E8 0000        CON(5)  =FOUR+          (ACCOUNT FOR TWO "'S)
          0
595 E61ED 0000        CON(5)  =LIT
          0
596 E61F2 FF00        CON(5)  255             EOS FF
          0
597 E61F7 0000        CON(5)  =SWAP           FF EOS
          0
598 E61FC 0000        CON(5)  =C!             empty stack
          0
599 E6201 0000        CON(5)  =SEMI
          0
600           ***********************************************
601           *
602           * Renamed SYNTAXF (from FSYNTAX)
603           * 12/18/83
604           *
605           * SYNTAXF       ( -- t/f )
606           * Checks for legal file specifier.
607           * String has already been put in STMTR0 by
608           * SETNAM.
609           * Stores blankfilled filename in variable oFLNAM.
610           *
611           ***********************************************
612 E6206 1F06        CON(5)  =!FCRT
          E
613 E620B 78   =!FSYNT CON(2)  #87
614 E620D 3595        NIBASC  \SYNTAX\
          E445
          1485
615 E6219 6C          CON(2)  \F\+#80
616 E621B 0000 =FSTX  CON(5)  =DOCOL
          0
617 E6220 6616        CON(5)  =SETNAM         SET UP THE NAME IN STMTR0
          E
618 E6225 F226        CON(5)  =FSYNTX         RETURN FLAG
```



```
          E
619 E622A 0000        CON(5)  =SEMI
          0
620
621 E622F 4326 =FSYNTX CON(5)  =$FSYNT
          E
622 E6234 8E00 =$FSYNT GOSUBL  =SAVEFP
          00
623 E623A 24           P=      4
624 E623C 8F00         GOSBVL  =R<RSTK
          000
625 E6243 1B00         D0=(5)  =STMTR0
          000
626 E624A 8F00         GOSBVL  =FSPECx         CHECK IT OUT
          000
627           *                                 IF LEGAL, CARRY CLEAR
628           *                                 AND NAME BLANKFILLED IN A
629 E6251 841          ST=0    1
630 E6254 4B0          GOC     FSYNT1
631 E6257 B47          D=D+1   S               D[S]=F IS NO DEVICE
632 E625A 550          GONC    FSYNT1
633 E625D 851          ST=1    1               FLAG GOOD
634 E6260 1B01 FSYNT1  D0=(5)  =oFNAME
          CF2
635 E6267 1507         DAT0=A  W               STORE IN VARIABLE
636 E626B 24           P=      4
637 E626D 8F00         GOSBVL  =RSTK<R
          000
638 E6274 8E00         GOSUBL  =GETFP
          00
639 E627A D2           C=0     A
640 E627C 861          ?ST=0   1               BAD RESULT?
641 E627F 40           GOYES   FSYNT2
642 E6281 CE           C=C-1   A               ELSE -1 FOR GOOD
643 E6283 1C4  FSYNT2  D1=D1-  5
644 E6286 145          DAT1=C  A               PUT FLAG ON STACK
645 E6289 03           RTNCC
646           ***********************************************
647           *
648           * CRFILE        ( size  fnameptr -- faddr/F )
649           * Make new file with given name and size.
650           * If desired size is negative, then we will create
651           * a 0 length file.
652           *
653           ***********************************************
654
655 E628B 0926 =CRFIL CON(5)  =$CRFL
          E
656 E6290 8E00 =$CRFL  GOSUBL  =SAVEFP
          00
657 E6296 143          A=DAT1  A               GET ADDR OF FILENAME
658 E6299 133          AD1EX
659 E629C 1577         C=DAT1  W               GET PROPER FILENAME
660 E62A0 1B00         D0=(5)  =STMTR0       where CRFSUB wants filename
          000
```

```
661 E62A7 1547         DAT0=C  W
662 E62AB 131          D1=A                restore FORTH stack
663 E62AE 174          D1=D1+  5             POINT TO SIZE OF FILE DESIRED
664 E62B1 143          A=DAT1  A             A=RECORD SIZE FOR CRFSUB
665 E62B4 D6           C=A     A             make copy of size in C(A)
666 E62B6 C6           C=C+C   A             shift left to test for neg #
667 E62B8 540          GONC    crf1          no carry means positive #
668 E62BB D0           A=0     A             set desired size to 0
669 E62BD 16F  crf1    D0=D0+  16          point to STMTR1,where type goes
670 E62C0 AF2          C=0     W
671 E62C3 304          LCHEX   4           copy code for text file
672 E62C6 816          CSRC                goes in [15]
673 E62C9 2A           P=      10
674 E62CB 301          LCHEX   1           type TEXT image in C
675 E62CE 20           P=      0
676 E62D0 1547         DAT0=C  W
677            *  get ready for CRETF+/CRFSB-
678 E62D4 23           P=      3
679 E62D6 8F00         GOSBVL  =R<RSTK
          000
680 E62DD D2           C=0     A
681 E62DF E6           C=C+1   A
682 E62E1 10A          R2=C                # of records  "   " (=1)
683 E62E4 20           P=      0
684 E62E6 3100         LC(2)   =oIMPLh     header size
685 E62EA C2           C=C+A   A           C=total size of file
686 E62EC AF3          D=0     W           D(S) must be 0 for CRETF+
687 E62EF 8F00         GOSBVL  =CRETF+
          000
688 E62F6 475          GOC     CRFL0       file not created,
689            *                           RETURN A FALSE FLAG
690
691 E62F9 8F00         GOSBVL  =CRFSB-     leave D1 pointing to file
          000
692            * PUT IN THE EOF MARKER (FFFF) IF FILE IS AT LEAST 2 BYTES LONG
693 E6300 133          AD1EX               save file ptr
694 E6303 20           P=      0
695 E6305 101          R1=A              save ptr in R1
696 E6308 131          D1=A
697 E630B 1C4          D1=D1-  5         point to length field
698 E630E 143          A=DAT1  A         A=offset to EOF from here
699 E6311 D2           C=0     A
700 E6313 3190         LC(2)   9         file of size 4 has 5+4=9
701 E6317 8B2          ?A<C    A         file<2 bytes?
702 E631A 21           GOYES   crf11
703 E631C 111          A=R1              restore ptr to first nib of file
704 E631F 131          D1=A
705 E6322 33FF         LCHEX   FFFF        EOF marker
          FF
706 E6328 15D3         DAT1=C  4           put it in
707 E632C 111  crf11   A=R1              restore file ptr
708 E632F D2           C=0     A
709 E6331 3100         LC(2)   =oIMPLh      FILE HEADER SIZE
710 E6335 EA           A=A-C   A            POINT TO START OF WHOLE FILE
711 E6337 23   CRFRES  P=      3
```



```
712 E6339 8F00         GOSBVL  =RSTK<R
          000
713 E6340 8E00         GOSUBL  =GETFP
          00
714 E6346 174          D1=D1+  5
715 E6349 141          DAT1=A  A
716 E634C 03           RTNCC
717
718 E634E D0   CRFL0   A=0     A
719 E6350 66EF         GOTO    CRFRES
```

```
                EJECT
              ****************************************
              *
              * COMPARE NEXT CHARACTER TO QUOTE (34)
              *
              ****************************************

E6354 00000 =NQOT? CON(5)  =DOCOL       :
E6359 43F5E        CON(5)  =STRM        'STREAM
E635E 00000        CON(5)  =CAT         O@
E6363 00000        CON(5)  =LIT
E6368 22000        CON(5)  34           \"\
E636D 00000        CON(5)  =EQUAL        =
E6372 00000        CON(5)  =SEMI         ;

              ************************************************
              *
              * Renamed ADJUSTF (from FADJUST)
              * 12/18/83
              *
              * ADJUSTF                    ( addr n -- F )
              *
              * Given amount to add (or subtract if n is
              * negative) and place to add (subtract), do it.
              * ((MVMEM+ subtracts below ptr, so in case of n=-4,
              * I first add 4 to addr.))
              * Code checks for addresses above or below whole file
              * chain, or pointing into the file header.
              * If addr is bad, errors out through ABORT; flag
              * is left to indicate other error condition, memory.
              *
              ************************************************

E6377 B026E        CON(5)  =!FSYNT
E637C 78   =!FADJ  CON(2)  #87
E637E 1444A455
      3545         NIBASC  \ADJUST\
E638A 6C           CON(2)  \F\+#80
E638C 00000 =FADJ  CON(5)  =DOCOL
E6391 2A36E        CON(5)  =FADJU
E6396 00000        CON(5)  =ABRT"X
E639B B2           CON(2)  =eRANGE
E639D 0000         CON(5)  =SEMI
```



```
      0

              *
              * DO ALL THE WORK, LEAVE FLAG FOR ABORTING
              *
E63A2 7A36E =FADJU CON(5)  =$FADJ
E63A7 143   =$FADJ A=DAT1  A              get n
E63AA 100          R0=A                   SAVE N
E63AD 174          D1=D1+  5
E63B0 147          C=DAT1  A              GET ADDR
E63B3 D7           D=C     A              D=ADDR
E63B5 174          D1=D1+  5              D1 SET IN FINAL POSITION
E63B8 8E00
      00           GOSUBL  =SAVEFP
              * NOW FIND THE FILE WE'RE IN
E63BE 1B00
      000          D0=(5)  =MAINST
E63C5 142          A=DAT0  A              GET START OF 1ST FILE
E63C8 D6           C=A     A              TO COMPARE
E63CA 8B7          ?C>D    A              ADDR BELOW FILE CHAIN?
E63CD D7           GOYES   rngerr
E63CF 101   SKIP   R1=A                   SAVE WHERE WE ARE
E63D2 8F00
      000          GOSBVL  =FILSK+        FIND NEXT FILE IN CHAIN
              *                            C[A]=NEXT FILE
E63D9 8BF          ?C>=D   A              NEXT FILE PAST OR AT ADDR?
E63DC 31           GOYES   FADJ01
              * NOTE THAT THIS MEANS AN ADDR OF START OF FILE N IS
              * TAKEN TO MEAN INSERT AT END OF FILE N-1
E63DE DA           A=C     A              SKIP ANOTHER
E63E0 134          D0=C
E63E3 14E          C=DAT0  B              GET 1ST BYTE OF NEXT FILE
E63E6 96A          ?C=0    B              ZERO MEANS END OF CHAIN
E63E9 16           GOYES   rngerr
E63EB 63EF         GOTO    SKIP
E63EF DB    FADJ01 C=D     A              GET ADDR
E63F1 DA           A=C     A              IT GOES IN A FOR MVMEM
E63F3 118          C=R0                   GET BACK N
E63F6 D5           B=C     A              N in B
E63F8 3400
      008          LCHEX   80000
E63FF 8B5          ?B<C    A              IS N POSITIVE?
E6402 80           GOYES   plus
              * for shrinking file:
E6404 D9           C=B     A              get amount to shrink
E6406 FA           C=-C    A              absolute value
E6408 CA           A=A+C   A              point ahead
              *
E640A 119   plus   C=R1                   filehead in C
              *                            addr in A
              * CHECK FOR ADDR WITHIN FILEHEADER
E640D 102          R2=A                SAVE
E6410 EA           A=A-C   A           HOW FAR PAST START OF FILE?
E6412 3400         LC(5)   =oIMPLh     LENGTH OF HEADER
```

```
          000
  811 E6419 8B2         ?A<C    A           IN HEADER?
  812 E641C E2          GOYES   rngerr
  813 E641E 112         A=R2
  814 E6421 119         C=R1                RESTORE
  815           *
  816 E6424 8F00        GOSBVL  =MVMEM+     expand or shrink,
          000
  817           *                           and update pointers
  818 E642B D3          D=0     A           FLAG INDICATING MEMERR
  819 E642D 440         GOC     nomove
  820 E6430 CF          D=D-1   A           FLAG OK
  821 E6432 D0    nomove A=0    A           FLAG FOR ABORT
  822 E6434 8E00 FADJX  GOSUBL  =GETFP
          00
  823 E643A 1C4         D1=D1-  5
  824 E643D DB          C=D     A          GET MEMERR FLAG
  825 E643F 145         DAT1=C  A
  826 E6442 1C4         D1=D1-  5
  827 E6445 141         DAT1=A  A           LEAVE FLAG
  828 E6448 03          RTNCC
  829 E644A D0    rngerr A=0    A
  830 E644C CC          A=A-1   A
  831 E644E 65EF        GOTO    FADJX
```



```
  832                 EJECT
  833           ****************************************
  834           *
  835           * START OF 8 CHAR WORDS
  836           * ADD ADDITIONAL WORDS AT THIS END
  837           *
  838           ****************************************
  839
  840           ****************************************
  841           *
  842           * PAGESIZE                ( -- N )
  843           *
  844           * System user variable used by assembler.
  845           *
  846           ****************************************
  847
  848 E6452 0000        CON(5)  =!ASSEM
          0
  849 E6457 88    =!PAGES CON(2)  #88
  850 E6459 0514        NIBASC  \PAGESIZ\
          7454
          3594
          A5
  851 E6467 5C          CON(2)  \E\+#80
  852 E6469 E646 =PAGESZ CON(5)  =$PAGES
          E
  853 E646E 8E00 =$PAGES GOSUBL  =DOUSE
          00
  854 E6474 84CF        CON(5)  (=oPAGES)
          2
```

```
                        EJECT
                *******************************************
                *
                * FLITERAL  ( ----  )  If compiling, then
                *       compile the value of the X register
                *       into the dictionary; else nothing.
                *
                *       Does not affect the integer stack.
                *
                *
                *       IMMEDIATE
                *******************************************
E6479 7546      E       CON(5) =!PAGES
E647E 8C        =!FLITR CON(2) #C8
E6480 64C4      9445    NIBASC \FLITERA\
              5425
              14
E648E CC                CON(2) \L\+#80

E6490 0000      =FLITR  CON(5) =DOCOL          :
              0
E6495 0000              CON(5) =STATE         STATE
              0
E649A 0000              CON(5) =AT            @
              0
E649F 68D5      E       CON(5) =ZBRNH         IF
E64A4 8200              CON(5) (FLIT0)-*
              0
E64A9 1DF5      E       CON(5) =COMPL          compile the f.p.
E64AE 0000              CON(5) =FLIT           literal handler into
              0
E64B3 0000              CON(5) =HERE           the dictionary followed
              0
E64B8 0000              CON(5) =STO            by the contents of the
              0
E64BD 0000              CON(5) =LIT            X register; then
              0
E64C2 0100              CON(5) 16              allot 16 nibs for that
              0
E64C7 0000              CON(5) =ALLOT          value
              0
E64CC 0000      FLIT0   CON(5) =SEMI           ;
              0
```



```
                        EJECT
                *******************************************
                *
                * CONSTANT   ( n - )   used as    31416 CONSTANT PI
                *
                *******************************************

E64D1 E746      E       CON(5)  =!FLITR        NO MORE WORDS
E64D6 88        =!CONST CON(2)  #88             CONSTANT
E64D8 34F4      E435    NIBASC  \CONSTAN\
              4514
              E4
E64E6 4D                CON(2)  \T\+#80

E64E8 0000      =CONST  CON(5) =DOCOL
              0
E64ED 0000              CON(5)  =CREAT         CREATE
              0
E64F2 0000              CON(5)  =COMMA         ,
              0
E64F7 0000              CON(5)  =ROM;C         ROM;CODE
              0
E64FC 0000              CON(5) =DOCON          RAM.doconstant
              0


                *******************************************
                *
                * VARIABLE   ( - )     used as    VARIABLE WINDSPEED
                * Allocates a 5 nibble variable.
                *
                *******************************************

E6501 6D46      E       CON(5)  !CONST
E6506 88        =!VAR   CON(2) #88             VARIABLE
E6508 6514      2594    NIBASC \VARIABL\
              1424
              C4
E6516 5C                CON(2) \E\+#80

E6518 0000      =VAR    CON(5) =DOCOL
              0
E651D 0000              CON(5)  =CREAT         CREATE
              0
E6522 0000              CON(5)  =LIT
              0
E6527 5000              CON(5) 5               5
              0
E652C 0000              CON(5)  =ALLOT         NALLOT
              0
E6531 0000              CON(5)  =SEMI          ;
```

0


```
                        EJECT
                *************************************************************
                *
                * EXPECT96  (addr --- ) NOTE: we use TITAN mainframe rtne
                *         CHEDIT to allow user entry of a string, thus we
                *         dispense with the expected char cnt.  We put 2
                *         nulls at the end of the string.
                *
                *        This rtne also implements the command stack,
                *        as well as user re-defined keys.
                *
                *       We use MAKEBF to process the
                *       raw input; it stores this input at RFNBFR it
                *       ends the input with a CR.  We can't use this
                *       space as our TERMINAL INPUT BUFFER because if
                *       we replace the CR with 1 or 2 NULLs we will
                *       screw up the system (according to B.Stephens)
                *       So we move the data from RFNBFR to our TIB
                *       and put 2 nulls at the end of the string (the
                *       CR is overwritten by one of the NULLs.
                *
                *************************************************************

E6536 6056E             CON(5)  =!VAR
E653B 88        =!EXPCT CON(2)  #88             EXPECT96
E653D 5485              NIBASC  \EXPECT9\
      0554
      3445
      93
E654B 6B                CON(2)  \6\+#80
E654D 2556E     =EXPCT  CON(5)  =$EXPCT

                * note: that we pretend to drop the addr on the data stack
                *       this saves code in the case where the user turns the
                *       machine off in this routine.  And if they do enter
                *       data its simply a D1=D1-5  statement later on in this
                *       rtne.

E6552 174       =$EXPCT D1=D1+  5
E6555 8E00              GOSUBL  =SAVEFP         SAVE FORTH POINTERS
      00
E655B 20                P=      0
E655D 1B00              D0=(5)  =NEEDSC
      000
E6564 31F0              LCHEX   0F
E6568 14C               DAT0=C  B               SAY YES, WE NEED TO SCROLL
                *                       OTHERWISE SCRLLR WON'T DO A THING
E656B 8F00      EXP99   GOSBVL  =SCRLLR         WAIT FOR KEY, DON'T
      000
                *                               LOSE CURRENT DISPLAY
E6572 968               ?A=0    B
E6575 6F                GOYES   EXP99           IGNORE TIMEOUT
E6577 8F00              GOSBVL  =CMD1ST         INIT CMD STACK POINTER
```

```
          000
  969
  970 E657E 8F00 EXP88   GOSBVL  =CHEDIT        GET USER ENTRY
          000
  971 E6585 4C1          GOC     EXP86          NOT A REDEFINED KEY!
  972
  973             *
  974             * WE'VE GOTTEN AN IMMEDIATE EXECUTE REDEFINED KEY
  975             * FROM THE USER.  A(A)=DEFINITION LEN+1 NIB
  976             * D1-->1ST CHAR OF DEFINITION
  977             *
  978 E6588 137          CD1EX
  979 E658B 108          R0=C                   SAVE ADDR IN R0
  980 E658E E4           A=A+1   A              EXP2 EXPECTS THE LENTH TO BE
  981 E6590 E4           A=A+1   A              4 GREATER THAN ACTUAL COUNT
  982 E6592 E4           A=A+1   A
  983 E6594 101          R1=A                   SAVE LENGTH COUNT+4
  984 E6597 8F00         GOSBVL  =FINLIN        FINISH LINE
          000
  985 E659E 6260         GOTO    EXP2           PROCESS DEFINITION
  986             *                             KEYS YET
  987 E65A2 8F00 EXP86   GOSBVL  =FINDA         FIND CHEDIT EXIT CONDITION
          000
  988 E65A9 D0           CON(2)  13             ENDLINE
  989 E65AB C20          REL(3)  EXP1
  990 E65AE 00           CON(2)  =kcUP          CURSOR UP KEY
  991 E65B0 7E0          REL(3)  INPUP
  992 E65B3 00           CON(2)  =kcDOWN        CURSOR DOWN KEY
  993 E65B5 BF0          REL(3)  INPDWN
  994 E65B8 00           CON(2)  =kcTOP         CURSOR TOP KEY
  995 E65BA 301          REL(3)  INPTOP
  996 E65BD 00           CON(2)  =kcBOT         CURSOR BOTTOM KEY
  997 E65BF EA0          REL(3)  INPBOT
  998 E65C2 00           CON(2)  =kcLAST        COMMAND STACK KEY
  999 E65C4 9A0          REL(3)  INPBOT
 1000 E65C7 00           CON(2)  =kcOFF         OFF KEY
 1001 E65C9 001          REL(3)  INPOFF
 1002 E65CC 00           CON(2)  =kcATTN
 1003 E65CE 680          REL(3)  INPATN
 1004 E65D1 00           CON(2)  0
 1005 E65D3 6AAF EXP0    GOTO    EXP88          GOT A STRANGE END CONDITION
 1006            *                              FOR NOW TRASH IT & RESTART CHEDIT
 1007
 1008 E65D7 8F00 EXP1    GOSBVL   =FINLIN        FINISH CURRENT LINE
          000
 1009 E65DE 8F00         GOSBVL  =MAKEBF        PROCESS RAW INPUT
          000
 1010            *                              RETURNS: C(A)=ADDR OF 1ST CHAR
 1011            *                              A(A)=LINE LENGTH+3(IN NIBS)
 1012            *                              (THIS INCLUDES THE CR)
 1013
 1014 E65E5 108          R0=C                   SAVE STRING START ADDR
 1015 E65E8 CC           A=A-1   A              NOW WE HAVE LENGTH WITH CR
 1016            *                              SINCE WE WANT TO ADD NULLS, IT'S
 1017 E65EA 101          R1=A                   OK TO BE 2 GREATER
```



```
 1018            *
 1019            * now we set the user variable SPAN to equal the number
 1020            * of characters read by EXPECT96 as required by FORTH 83
 1021            * Standard
 1022            *
 1023 E65ED 1BC5         D0=(5)  =oSPAN         GET ADDR TO D0
          BF2
 1024 E65F4 AD0          A=0     M              PROTECT AGAINST STRAY BITS
 1025 E65F7 81C          ASRB                   IN SHIFT, MAKE NIBS=BYTES
 1026 E65FA CC           A=A-1   A              GET RID OF EXTRA CHAR CNT
 1027 E65FC CC           A=A-1   A              AND COUNT OF CR
 1028 E65FE 140          DAT0=A  A              WRITE OUT COUNT
 1029
 1030            *       LC(2)   \E\            RESET DISPLAY
 1031            *       GOSBVL  =ESCSEQ
 1032
 1033 E6601 8E00 EXP2    GOSUBL  =GETFP         RECOVER FORTH POINTERS, DOESN'T
          00
 1034            *                              TRASH A(A) OR D(A)
 1035 E6607 1C4          D1=D1-  5              RECOVER ADDR
 1036            *
 1037            * SEE ABOVE FOR WHY WE PRETEND TO DROP ADDR
 1038            *
 1039 E660A 147          C=DAT1  A              DESTINATION ADDR
 1040 E660D 1C4          D1=D1-  5
 1041 E6610 111          A=R1                   RECOVER LENGTH
 1042 E6613 141          DAT1=A  A              PUSH CHAR CNT + 2 (NIBS)
 1043 E6616 D7           D=C     A              SAVE DEST ADDR
 1044 E6618 118          C=R0                   C=SOURCE ADDR
 1045 E661B 1C4          D1=D1-  5
 1046 E661E 145          DAT1=C  A              PUSH SOURCE ADDR
 1047 E6621 DB           C=D     A
 1048 E6623 1C4          D1=D1-  5
 1049 E6626 145          DAT1=C  A              PUSH DEST ADDR
 1050 E6629 1C4          D1=D1-  5
 1051 E662C 141          DAT1=A  A              PUSH STRING LENGTH(IN NIBS)
 1052
 1053 E662F 8E00         GOSUBL  =$NMOVE        MOVE STRING TO TIB
          00
 1054 E6635 143          A=DAT1  A              POP COUNT
 1055 E6638 174          D1=D1+  5
 1056 E663B 147          C=DAT1  A              POP STRING ADDR
 1057 E663E 174          D1=D1+  5              RETURN NOTHING ON STACK
 1058 E6641 C2           C=C+A   A              ADDR OF STR END + 2
 1059 E6643 137          CD1EX
 1060 E6646 1C3          D1=D1-  4              OVER WRITE CR
 1061 E6649 D0           A=0     A
 1062 E664B 1593         DAT1=A  4              WRITE 2 NULLS
 1063 E664F 135          D1=C                   RESTORE DATA STK PTR
 1064 E6652 03           RTNCC
 1065
 1066 E6654 8F00 INPATN  GOSBVL =ATNCLR         CLEAR ATTN FLAG
          000
 1067 E665B 8F00         GOSBVL  =CURSFL        SEND CURSOR TO FAR LEFT
          000
```

```
1068 E6662 8F00          GOSBVL =-LINE          CLEAR LINE
           000
1069 E6669 641F          GOTO   EXP88           GET MORE INPUT
1070
1071 E666D 8F00 INPBOT   GOSBVL =CMD1ST         INIT COMMAND STACK
           000
1072
1073 E6674 8F00 INPB10   GOSBVL  =FINLIN         FINISH CURRENT LINE
           000
1074 E667B 8F00          GOSBVL =CMDFND         FIND BUFFER
           000
1075 E6682 172           D1=D1+ 3
1076 E6685 137           CD1EX                  C PTS TO BUFF START
1077 E6688 7E50          GOSUB  GPRMPT          SET UP WITH CMD STK "\"
1078 E668C 8F00          GOSBVL =REPROM         REPROMPT USER
           000
1079 E6693 6AEE          GOTO   EXP88
1080
1081 E6697 8F00 INPUP    GOSBVL =CMDINI         READ CURRENT & MAX CMD
           000
1082 E669E B04           A=A+1  P               INCREMENT CMD NUMBER
1083 E66A1 42D           GOC    INPB10          WATCH FOR OVERFLOW
1084 E66A4 986  INPU10   ?A>C   P              GT MAX CMD?
1085 E66A7 DC            GOYES  INPB10          YES, DON'T CHANGE IT
1086 E66A9 1590 INPU20   DAT1=A 1              CHANGE CMD NUMBER
1087 E66AD 56C           GONC   INPB10          B.E.T.
1088
1089 E66B0 8F00 INPDWN   GOSBVL =CMDINI
           000
1090 E66B7 CC            A=A-1  A
1091 E66B9 6AEF          GOTO   INPU10
1092
1093 E66BD 8F00 INPTOP   GOSBVL =CMDINI
           000
1094 E66C4 DA            A=C    A               SET CMD NUMBER TO MAX
1095 E66C6 52E           GONC   INPU20          B.E.T.
1096
1097 E66C9 1BC0 INPOFF   D0=(5)  =ACTIVE
           BF2
1098 E66D0 20            P=     0               write a 99 to ACTIVE,
1099 E66D2 3436          LC(5)  99              this is the mainloop
           000
1100 E66D9 144           DAT0=C A               poll handler`s clue to
1101             *                              restart in FORTH
1102 E66DC 8F00          GOSBVL =FINLIN         finish line
           000
1103 E66E3 8D00          GOVLNG =PWROFF         put machine to deep sleep
           000
1104
1105
1106 E66EA 10B  GPRMPT   R3=C
1107 E66ED 7900          GOSUB  GPR0
1108 E66F1 22            CON(2) \"\
1109 E66F3 C5            CON(2) 92            USE "\" AS PROMPT!
1110 E66F5 22            CON(2) \"\
```



```
1111 E66F7 000           CON(3)  0
1112
1113 E66FA 07   GPR0     C=RSTK                 C--> PROMPT
1114 E66FC 12B           CR3EX                  R3-->PROMPT, C(A)-->BUFFER
1115 E66FF 01            RTN
```

```
1116                    EJECT
1117              ********************************
1118              *
1119              * D?NEGATE  (d1 n --- d2)
1120              *
1121              * MADE HEADERLESS 10/10/83
1122              *
1123              ********************************
1124
1125 E6701 0000 =D?NEG  CON(5)  =DOCOL          :
          0
1126 E6706 0000         CON(5)   =LTZ           0<
          0
1127 E670B 68D5         CON(5)   =ZBRNH         IF
          E
1128 E6710 A000         CON(5)  (D?NO)-*
          0
1129 E6715 60D5         CON(5)   =DNGAT         DNEGATE
          E
1130               *                            THEN
1131 E671A 0000 D?NO    CON(5)   =SEMI          ;
          0
```



```
1132                    EJECT
1133              ********************************************************************
1134              *
1135              * TRAVERSE  (addr1 n --- addr2) Move across name field of a
1136              *       variable length dictionary header.  Addr1 is the addr
1137              *       of either the length byte or the last letter.  If n=1
1138              *       move toward high memory, if n=-1 move toward low memory.
1139              *
1140              *       If n is not 1 or -1 then return addr2=addr1!
1141              *
1142              ********************************************************************
1143
1144 E671F B356         CON(5)  !EXPCT
          E
1145 E6724 88   =!TRAVS CON(2)  #88             TRAVERSE
1146 E6726 4525         NIBASC  \TRAVERS\
          1465
          5425
          35
1147 E6734 5C           CON(2)  \E\+#80
1148
1149 E6736 0000 =TRAVS  CON(5)  =DOCOL          :
          0
1150 E673B 0000         CON(5)  =DUP            (addr1 n n)
          0
1151 E6740 0000         CON(5)  =ABS            (addr1 n |n|)
          0
1152 E6745 0000         CON(5)  =ONE            (addr1 n |n| 1)
          0
1153 E674A 0000         CON(5)  =EQUAL          (addr1 n t/f)
          0
1154 E674F 68D5         CON(5)  =ZBRNH          if n is not 1 or -1 quit
          E
1155 E6754 1400         CON(5)  (TRAV0)-*
          0
1156 E6759 0000         CON(5)   =TWO*          2*   BYTES TO NIBBLES
          0
1157 E675E 0000         CON(5)   =SWAP          SWAP
          0
1158 E6763 0000 TRA     CON(5)   =OVER          BEGIN OVER
          0
1159 E6768 0000         CON(5)   =ADD           +
          0
1160 E676D 0000         CON(5)   =LIT
          0
1161 E6772 F700         CON(5)  #7F             TEST FOR HIGH BIT SET
          0
1162 E6777 0000         CON(5)   =OVER          OVER
          0
1163 E677C 0000         CON(5)   =CAT           C@     GET LETTER
          0
1164 E6781 0000         CON(5)   =LT            <
          0
1165 E6786 68D5         CON(5)   =ZBRNH         UNTIL
          E
```

```
1166 E678B 8DFF          CON(5)  (TRA)-*
           F
1167 E6790 0000          CON(5)   =SWAP          SWAP
           0
1168 E6795 0000 TRAV0    CON(5)   =DROP          DROP
           0
1169 E679A 0000          CON(5)   =SEMI          ;
           0
```



```
1170                EJECT
1171           ************************************************************
1172           *
1173           * DLITERAL (d --- d ) executing  (d --- ) compiling
1174           *       If compiling, compile double # into a literal, if
1175           *       executing, nothing.
1176           *
1177           *        IMMEDIATE
1178           *
1179           ************************************************************
1180
1181 E679F 4276          CON(5)  =!TRAVS
           E
1182 E67A4 8C    =!DLITR CON(2)  #C8             DLITERAL
1183 E67A6 44C4          NIBASC  \DLITERA\
           9445
           5425
           14
1184 E67B4 CC            CON(2)  \L\+#80
1185 E67B6 0000  =DLITR  CON(5)  =DOCOL          :
           0
1186 E67BB 0000          CON(5)  =STATE         STATE
           0
1187 E67C0 0000          CON(5)  =AT            @
           0
1188 E67C5 68D5          CON(5)  =ZBRNH         IF
           E
1189 E67CA 4100          CON(5)  (DLIT0)-*
           0
1190 E67CF 0000          CON(5)   =SWAP           SWAP
           0
1191 E67D4 F8F5          CON(5)   =LITR           LITERAL
           E
1192 E67D9 F8F5          CON(5)   =LITR           LITERAL
           E
1193 E67DE 0000 DLIT0    CON(5)   =SEMI           ;
           0
1194
1195           ************************************************************
1196           *
1197           * CLOSEALL ( --- )  This routine is used to close all open
1198           * files in the fib area.  It is used by ABORT and ABORT"
1199           *
1200           ************************************************************
1201
1202 E67E3 4A76          CON(5)  =!DLITR
           E
1203 E67E8 88    =!CLALL CON(2)  #88             CLOSEALL (FILES)
1204 E67EA 34C4        NIBASC  \CLOSEAL\
           F435
           5414
           C4
1205 E67F8 CC          CON(2)  \L\+#80
1206
1207 E67FA FF76 =CLALL  CON(5) =$CLALL
```

```
            E
   1208
   1209 E67FF 8E00 =$CLALL GOSUBL  =SAVEFP         save forth pointers
            00
   1210 E6805 20           P=      0
   1211 E6807 8F00         GOSBVL  =CLOSEA         system rtne to close all files
            000
   1212 E680E 8C00         GOLONG  =GETFP          restore forth pointers
            00
   1213
   1214 E6814      =REND5  END
```



```
 !ASSEM  Ext                    -   848
=!CLALL  Abs  944104 #E67E8  -  1203
=!COMPL  Abs  942017 #E5FC1  -   421   475
=!CONST  Abs  943318 #E64D6  -   893   911
=!CONTX  Abs  941377 #E5D41  -   163   180
=!CONVT  Abs  941556 #E5DF4  -   251   294
=!CURR   Abs  941414 #E5D66  -   181   229
=!DEC    Abs  941505 #E5DC1  -   230   250
=!DEGR   Abs  941253 #E5CC5  -   108   123
=!DLITR  Abs  944036 #E67A4  -  1182  1202
=!DNGAT  Abs  941302 #E5CF6  -   124   162
=!ENCL   Abs  941702 #E5E86  -   295   363
=!ENDCS  Abs  942158 #E604E  -   476   525
=!EXEC   Abs  941166 #E5C6E  -    78    92
=!EXPCT  Abs  943419 #E653B  -   946  1144
=!FADJ   Abs  942972 #E637C  -   754
=!FCRT   Abs  942321 #E60F1  -   526   612
=!FLITR  Abs  943230 #E647E  -   868   892
=!FSYNT  Abs  942603 #E620B  -   613   753
=!LITR   Abs  941951 #E5F7F  -   393   420
=!LSTNG  Abs  941061 #E5C05  -    27    60
=!PAGES  Abs  943191 #E6457  -   849   867
=!PRIME  Abs  941129 #E5C49  -    61    77
=!RAD    Abs  941204 #E5C94  -    93   107
=!STRM   Abs  941860 #E5F24  -   364   392
=!TRAVS  Abs  943908 #E6724  -  1145  1181
=!VAR    Abs  943366 #E6506  -   912   945
=$BRNCH  Abs  941470 #E5D9E  -   212   208   210
=$CLALL  Abs  944127 #E67FF  -  1209  1207
=$CONTX  Abs  941398 #E5D56  -   168   166
=$CRFL   Abs  942736 #E6290  -   656   655
=$CURR   Abs  941435 #E5D7B  -   186   184
=$DEGR   Abs  941274 #E5CDA  -   112   111
=$DNGAT  Abs  941323 #E5D0B  -   129   127
=$ENCL   Abs  941723 #E5E9B  -   300   298
 $EXEC   Abs  941187 #E5C83  -    83    81
=$EXPCT  Abs  943442 #E6552  -   957   949
=$FADJ   Abs  943015 #E63A7  -   767   766
=$FNAM   Abs  942427 #E615B  -   554   553
=$FSYNT  Abs  942644 #E6234  -   622   621
=$LSTNG  Abs  941082 #E5C1A  -    31    30
 $NMOVE  Ext                    1053
=$PAGES  Abs  943214 #E646E  -   853   852
=$PRIME  Abs  941150 #E5C5E  -    66    64
=$RAD    Abs  941225 #E5CA9  -    97    96
=$RSIZE  Abs  941361 #E5D31  -   150   149
=$ZBRNH  Abs  941451 #E5D8B  -   204   202
 -LINE   Ext                    1068
 0B0     Abs  941489 #E5DB1  -   219   207
 2DROP   Ext                    536
 2DUP    Ext                    532
=<;COD   Abs  942108 #E601C  -   459
 <>      Ext                    486
 >R      Ext                    258   430   579
 ?COMP   Ext                    426   481
```

?DUP	Ext		-	371							
=?NEG	Abs	942078	#E5FFE -	443							
ABRT"X	Ext		-	487	759						
ABS	Ext		-	568	1151						
ADD	Ext		-	380	578	1159					
ALLOT	Ext		-	883	920						
AT	Ext		-	261	267	272	370	379	399	431	495
				874	1187						
ATNCLR	Ext		-	1066							
BASE	Ext		-	237	260	266	271				
BLK	Ext		-	369							
BLOCK	Ext		-	374							
=BRNCH	Abs	941465	#E5D99 -	210	277	375	510				
C!	Ext		-	584	592	598					
CAT	Ext		-	259	729	1163					
CHEDIT	Ext		-	970							
=CLALL	Abs	944122	#E67FA -	1207							
CLOSEA	Ext		-	1211							
CMD1ST	Ext		-	968	1071						
CMDFND	Ext		-	1074							
CMDINI	Ext		-	1081	1089	1093					
COMMA	Ext		-	404	432	899					
=COMPL	Abs	942033	#E5FD1 -	425	402	489	877				
=CONST	Abs	943336	#E64E8 -	897							
=CONTX	Abs	941393	#E5D51 -	166							
CONV1	Abs	941577	#E5E09 -	256	278						
CONV3	Abs	941687	#E5E77 -	279	264						
=CONVT	Abs	941572	#E5E04 -	255							
CREAT	Ext		-	898	917						
CRETF+	Ext		-	687							
=CRFIL	Abs	942731	#E628B -	655	545						
CRFL0	Abs	942926	#E634E -	718	688						
CRFRES	Abs	942903	#E6337 -	711	719						
CRFSB-	Ext		-	691							
CSP	Ext		-	494							
=CURR	Abs	941430	#E5D76 -	184							
CURSFL	Ext		-	1067							
D+	Ext		-	274							
D?N0	Abs	943898	#E671A -	1131	1128						
=D?NEG	Abs	943873	#E6701 -	1125							
=DEC	Abs	941521	#E5DD1 -	234							
=DEGR	Abs	941269	#E5CD5 -	111							
DIGIT	Ext		-	262							
DLIT0	Abs	944094	#E67DE -	1193	1189						
=DLITR	Abs	944054	#E67B6 -	1185							
=DNGAT	Abs	941318	#E5D06 -	127	1129						
DOATN	Ext		-	216	220						
DOCOL	Ext		-	234	255	368	397	425	443	459	480
				529	567	616	727	757	872	897	916
				1125	1149	1185					
DOCON	Ext		-	901							
DOUSE	Ext		-	66	150	168	186	554	853		
DROP	Ext		-	269	490	506	537	1168			
DUP	Ext		-	257	428	482	500	572	576	1150	
=ENCL	Abs	941718	#E5E96 -	298							

ENCL1	Abs	941755	#E5EBB -	312	317						
ENCL2	Abs	941797	#E5EE5 -	330	322						
ENCL20	Abs	941799	#E5EE7 -	331	338						
ENCL3	Abs	941827	#E5F03 -	342	336						
ENCL4	Abs	941836	#E5F0C -	346	329	341					
ENDC1	Abs	942221	#E608D -	493	511						
ENDC2	Abs	942256	#E60B0 -	500	498						
ENDC3	Abs	942311	#E60E7 -	512	505						
=ENDCS	Abs	942174	#E605E -	480							
EQUAL	Ext		-	496	503	732	1153				
=EXEC	Abs	941182	#E5C7E -	81							
EXP0	Abs	943571	#E65D3 -	1005							
EXP1	Abs	943575	#E65D7 -	1008	989						
EXP2	Abs	943617	#E6601 -	1033	985						
EXP86	Abs	943522	#E65A2 -	987	971						
EXP88	Abs	943486	#E657E -	970	1005	1069	1079				
EXP99	Abs	943467	#E656B -	964	967						
=EXPCT	Abs	943437	#E654D -	949							
=FADJ	Abs	942988	#E638C -	757							
FADJ01	Abs	943087	#E63EF -	793	784						
=FADJU	Abs	943010	#E63A2 -	766	758						
FADJX	Abs	943156	#E6434 -	822	831						
FCFAIL	Abs	942372	#E6124 -	536							
FCR01	Abs	942377	#E6129 -	537	543						
FCR02	Abs	942392	#E6138 -	541	535						
=FCREAT	Abs	942337	#E6101 -	529							
FFIND	Ext		-	533							
FILSK+	Ext		-	781							
FINDA	Ext		-	987							
FINLIN	Ext		-	984	1008	1073	1102				
FIVE+	Ext		-	429							
FLIT	Ext		-	878							
FLIT0	Abs	943308	#E64CC -	884	876						
=FLITR	Abs	943248	#E6490 -	872							
=FNAME	Abs	942422	#E6156 -	553	544						
FOUR+	Ext		-	594							
FSPECx	Ext		-	626							
=FSTX	Abs	942619	#E621B -	616	541						
FSYNT1	Abs	942688	#E6260 -	634	630	632					
FSYNT2	Abs	942723	#E6283 -	643	641						
=FSYNTX	Abs	942639	#E622F -	621	618						
GETFP	Ext		-	100	115	638	713	822	1033	1212	
GPR0	Abs	943866	#E66FA -	1113	1107						
GPRMPT	Abs	943850	#E66EA -	1106	1077						
HERE	Ext		-	879							
IN	Ext		-	378							
INPATN	Abs	943700	#E6654 -	1066	1003						
INPB10	Abs	943732	#E6674 -	1073	1083	1085	1087				
INPBOT	Abs	943725	#E666D -	1071	997	999					
INPDWN	Abs	943792	#E66B0 -	1089	993						
INPOFF	Abs	943817	#E66C9 -	1097	1001						
INPTOP	Abs	943805	#E66BD -	1093	995						
INPU10	Abs	943780	#E66A4 -	1084	1091						
INPU20	Abs	943785	#E66A9 -	1086	1095						
INPUP	Abs	943767	#E6697 -	1081	991						

LATE	Ext		-	461								
=LISTNG	Abs	941077	#E5C15	-	30							
LIT	Ext		-	235	403	463	483	501	569	574	580	
				589	595	730	881	918	1160			
=LITR	Abs	941967	#E5F8F	-	397	1191	1192					
LITR0	Abs	942007	#E5FB7	-	405	401						
LT	Ext			-	1164							
LTZ	Ext			-	444	1126						
MAINST	Ext			-	775							
MAKEBF	Ext			-	1009							
MIN	Ext			-	571							
MINUS	Ext			-	465							
MVMEM+	Ext			-	816							
NEEDSC	Ext			-	960							
NEGAT	Ext			-	447							
NG0	Abs	942103	#E6017	-	448	446						
=NQOT?	Abs	942932	#E6354	-	727							
ONE	Ext			-	1152							
OVER	Ext			-	582	1158	1162					
=PAGESZ	Abs	943209	#E6469	-	852							
PFA	Ext			-	462							
=PRIME	Abs	941145	#E5C59	-	64							
PWROFF	Ext			-	1103							
R<RSTK	Ext			-	624	679						
R>	Ext			-	275	279	427	460	593			
R@	Ext			-	587							
=RAD	Abs	941220	#E5CA4	-	96							
=REND5	Abs	944148	#E6814	-	1214							
REPROM	Ext			-	1078							
ROM;C	Ext			-	900							
ROT	Ext			-	270	530	531	577				
=RSIZE	Abs	941356	#E5D2C	-	149							
RSTK<R	Ext			-	637	712						
RUN	Ext			-	85							
SAVEFP	Ext			-	97	112	622	656	773	958	1209	
SCRLLR	Ext			-	964							
SEMI	Ext			-	239	280	381	405	433	448	467	499
					512	539	546	599	619	733	761	884
					921	1131	1169	1193				
=SETNAM	Abs	942438	#E6166	-	567	617						
SFLAGC	Ext			-	114							
SFLAGS	Ext			-	99							
SKIP	Abs	943055	#E63CF	-	780	792						
SMOVE	Ext			-	586							
SP@	Ext			-	493							
ST0	Abs	941921	#E5F61	-	377	373						
ST01	Abs	941926	#E5F66	-	378	376						
STATE	Ext			-	398	873	1186					
STMIR0	Ext			-	575	625	660					
STO	Ext			-	880							
STORE	Ext			-	238	466						
=STRM	Abs	941876	#E5F34	-	368	728						
SWAP	Ext			-	265	591	597	1157	1167	1190		
THEN	Ext			-	509							
TIB	Ext			-	377							

TRA	Abs	943971	#E6763	-	1158	1166						
TRAV0	Abs	944021	#E6795	-	1168	1155						
=TRAVS	Abs	943926	#E6736	-	1149							
TWO	Ext			-	507							
TWO*	Ext			-	573	1156						
TWO+	Ext			-	256	585	588					
U*	Ext			-	268	273						
=VAR	Abs	943384	#E6518	-	916							
=ZBRNH	Abs	941446	#E5D86	-	202	263	372	400	445	497	504	534
					542	875	1127	1154	1165	1188		
ZERO	Ext			-	538							
crf1	Abs	942781	#E62BD	-	669	667						
crf11	Abs	942892	#E632C	-	707	702						
f1RAD	Ext			-	98	113						
kcATTN	Ext			-	1002							
kcBOT	Ext			-	996							
kcDOWN	Ext			-	992							
kcLAST	Ext			-	998							
kcOFF	Ext			-	1000							
kcTOP	Ext			-	994							
kcUP	Ext			-	990							
nomove	Abs	943154	#E6432	-	821	819						
oIMPLh	Ext			-	684	709	810					
plus	Abs	943114	#E640A	-	805	799						
rngerr	Abs	943178	#E644A	-	829	779	791	812				

Input Parameters

Source file name is MR&FT5

Listing file name is MR/FT5::65

Object file name is MR%FT5::65

```
                                  111111
                        0123456789012345
```

Initial flag settings are

Errors

None

Saturn Assembler News

```
                TITLE  FT6:_9_10_etc._CHAR WORDS
                RDSYMB MR%GTO
*               RDSYMB MR%FT5
*               ABS    REND5

E6B00           ABS    #E6B00

*************************************************
*
* ?TERMINAL   ( --- t/f)  This word checks to see
*              if a key has been hit; if it has then
*              it returns a true, else a false.
*
* NOTE:  once a key has been hit ?TERMINAL will
*        continue to return true until the key is
*        read with KEY
*
*************************************************
E6B00 0000      CON(5)  0
      0
E6B05 98   =!?TERM CON(2)  #89
E6B07 F345      NIBASC  \?TERMINA\
      5425
      D494
      E414
E6B17 CC        CON(2)  \L\+#80
E6B19 E1B6 =?TERM  CON(5)  =$?TERM
      E
E6B1E 136  =$?TERM CD0EX                  save I ptr in C
E6B21 1B00      D0=(5)  =KEYPTR           ptr to count of keys in buffer
      000
E6B28 1522      A=DAT0  XS                read key count
E6B2C 136       CD0EX                     restore I to D0
E6B2F D2        C=0     A                 set C(A) up as FALSE flag
E6B31 A2C       A=A-1   XS                ANYTHING IN KEY BUFFER?
E6B34 440       GOC     TERMX             CARRY SET IF NO
E6B37 CE        C=C-1   A                 make it TRUE
E6B39 1C4  TERMX   D1=D1-  5
E6B3C 145       DAT1=C  A                 push flag to stack
E6B3F 03        RTNCC
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER


```
                EJECT
*************************************************
*
* FCONSTANT    FORM:  (X) FCONSTANT  <name>
*
*       Create a floating point constant called
*       <name> which contains the current value
*       of the X register.  when <name> is
*       executed later the value is pushed onto
*       the floating point stack into the X
*       register.
*
*************************************************
E6B41 50B6      CON(5)  =!?TERM
      E
E6B46 98   =!FCON  CON(2)  #89
E6B48 6434      NIBASC  \FCONSTAN\
      F4E4
      3545
      14E4
E6B58 4D        CON(2)  \T\+#80

E6B5A 0000 =FCON   CON(5)  =DOCOL
      0
E6B5F 0000      CON(5)  =CREAT        CREATE ENTRY
      0
E6B64 0000      CON(5)  =HERE         PLACE TO PUT IT
      0
E6B69 0000      CON(5)  =STO          STORE VALUE
      0
E6B6E 0000      CON(5)  =LIT
      0
E6B73 0100      CON(5)  16            ALLOT 16 NIBS FOR FLOATING
      0
E6B78 0000      CON(5)  =ALLOT        POINT CONSTANT
      0
E6B7D 0000      CON(5)  =ROM;C        COMPILE FLOATING POINT
      0
E6B82 0000      CON(5)  =DOFCON       CONSTANT PROLOGUE
      0
```

```
                        EJECT
                    ************************************************
                    *
                    * FVARIABLE     FORM:  FVARIABLE <name>
                    *
                    *       create a floating point variable in the
                    *       user's dictionary called <name>.
                    *
                    ************************************************
E6B87 64B6E         =!FCON  CON(5)  !FCON
E6B8C 98            =!FVAR  CON(2)  #89
E6B8E 6465          =FVARIABL       NIBASC  \FVARIABL\
      1425
      9414
      24C4
E6B9E 5C                    CON(2)  \E\+#80

E6BA0 00000         =FVAR   CON(5)  =DOCOL
E6BA5 00000                 CON(5)  =CREAT      CREATE ENTRY CALLED
E6BAA 00000                 CON(5)  =LIT        <name> AND ALLOT
E6BAF 01000                 CON(5)  16          16 NIBS FOR VALUE
E6BB4 00000                 CON(5)  =ALLOT
E6BB9 00000                 CON(5)  =SEMI
```



```
                        EJECT
                    ************************************************
                    *
                    * HPIL-LOOP  ( --- addr ) Word which returns the addr
                    *       of a variable which contains the HPIL LOOP
                    *       number (1,2 or 3) over which data is to
                    *       be ENTERed and OUTPUT.
                    *
                    *       THIS WORD IS NOT AVAILABLE TO USER IN
                    *       CURRENT IMPLEMENTATION.
                    ************************************************

                    *       CON(5)  !FVAR
                    *=!HPLP  CON(2)  #89
                    *       NIBASC  \HPIL-LOO\
                    *       CON(2)  \P\+#80
E6BBE 3CB6E         =HPLP   CON(5)  =$HPLP

E6BC3 8E000         =$HPLP  GOSUBL  =DOUSE
E6BC9 7ABF2                 CON(5)  (=oLOOP)
```

```
                        EJECT
                *************************************************
                *
                * SECONDARY     ( --- addr ) Word which returns the addr
                *               of a variable which contains the HPIL
                *               SECONDARY addr (if any).
                *
                *************************************************

E6BCE C8B6E             CON(5)  =!FVAR
E6BD3 98        =!SECD  CON(2)  #89
E6BD5 3554 34F4 E444 1425
                        NIBASC  \SECONDAR\
E6BE5 9D                CON(2)  \Y\+#80
E6BE7 CEB6E     =SECD   CON(5)  =$SECD

E6BEC 8E0000    =$SECD  GOSUBL  =DOUSE
E6BF2 CABF2             CON(5)  (=oSECD)
```



```
                        EJECT
                *************************************************
                *
                * [COMPILE] Used in colon-defs to force compilation of
                *           the following word.  Allows compilation of
                * an IMMEDIATE word when it would otherwise be executed.
                *
                * THIS IS AN IMMEDIATE WORD
                *
                *************************************************

E6BF7 3DB6E             CON(5)  =!SECD          NO MORE 9 CHAR WORDS
E6BFC 9C        =![CMP] CON(2)  #C9             [COMPILE]
E6BFE B534 F4D4 0594 C454
                        NIBASC  \[COMPILE\
E6C0E DD                CON(2)  \]\+#80

E6C10 00000     =[CMP]  CON(5)  =DOCOL          :
E6C15 00000             CON(5)   =?COMP         ?COMP  error if not compiling
E6C1A 00000             CON(5)   =-FIND         -FIND  find tic addr of word
                *                                      -FIND rtns a flag to
                *                                      signal if word was found
E6C1F 00000             CON(5)   =ZEQ           0=     change sense of flag
E6C24 00000             CON(5)   =ABORTW        ABORT" abort if word not found
E6C29 20                CON(2)  =eNOTF          with "not found" message
E6C2B 00000             CON(5)   =DROP          DROP   drop char count of word
E6C30 00000             CON(5)   =COMMA         ,      compile tic addr
E6C35 00000             CON(5)   =SEMI          ;
```

```
                        EJECT
                *****************************************************************
                *
                * INTERPRET (---)  Begin interpretation at the char indexed
                *          by the contents of >IN relative to the block number
                *          contained in BLK, continuing until the input stream
                *          is exhausted.  If BLK=0 interpret chars from the
                *          terminal input buffer.
                *
                * VECTORED WORD
                *****************************************************************

E6C3A CFB6E     =!INTRP CON(2)  #89             INTERPRET
E6C41 94E44554  2505 2554
                        NIBASC  \INTERPRE\
E6C51 4D                CON(2)  \T\+#80

E6C53 00000     =INTRP  CON(5)  =DOCOL          :
E6C58 00000             CON(5)  =LIT
E6C5D 02CF2             CON(5)  =oINTRP         LOCATION OF INTERPRET VECTOR
E6C62 00000             CON(5)  =AT             GET IT
E6C67 00000             CON(5)  =EXEC           EXECUTE IT
E6C6C 00000             CON(5)  =SEMI

                *
                * DEFAULT INTERPRET
                *

E6C71 00000     =!INTRP CON(5)   =DOCOL         :
E6C76 00000     INT00   CON(5)   =-FIND          -FIND   attempt to find word in
E6C7B 00000             CON(5)   =ZBRNH          IF      dictionary, if found
E6C80 C3000             CON(5)  (INT0)-*                 check STATE.  If content
E6C85 00000             CON(5)   =STATE          STATE   of STATE are > then the
E6C8A 00000             CON(5)   =AT             @       raw length of word then
E6C8F 00000             CON(5)   =LT             <       the word is not immediat
E6C94 00000             CON(5)   =ZBRNH          IF      so compile it (remember
E6C99 41000             CON(5)  (INT1)-*                 STATE=0 or  #C0)
```



```
E6C9E 00000             CON(5)  =COMMA          ,       if raw length > STATE
E6CA3 00000             CON(5)  =BRNCH          ELSE    then we have an immediat
E6CA8 ECFFF             CON(5)  (INT00)-*               word so lets execute it
E6CAD 00000 INT1        CON(5)  =EXEC           EXECUTE

E6CB2 00000 INT2        CON(5)  =BRNCH          THEN THEN ELSE
E6CB7 FBFFF             CON(5)  (INT00)-*       START OVER AGAIN!

E6CBC 00000 INT0        CON(5)  =HERE           HERE  we couldn't find the word
E6CC1 00000             CON(5)  =NUMB           NUMBER so let's see if its a leg
                *                               NUMBER will error out if not a l
                *                               integer or floating point number
E6CC6 00000             CON(5)  =DPL            DPL   number
E6CCB 00000             CON(5)  =AT             @
E6CD0 00000             CON(5)  =DUP            DUP contents of DPL
E6CD5 00000             CON(5)  =LIT
E6CDA 00000             CON(5)  #C0000          see if we got a f.p.
E6CDF 00000             CON(5)  =EQUAL          number
E6CE4 00000             CON(5)  =ZEQ            0=  reverse sense of test
E6CE9 00000             CON(5)  =ZBRNH          branch if f.p. number
E6CEE 32000             CON(5)  (INT3)-*
E6CF3 00000             CON(5)  =ONE+           1+  see if DPL was #FFFFF
E6CF8 00000             CON(5)  =ZBRNH          IF if so it was a single number
E6CFD 82000             CON(5)  (INT4)-*        else it was a double number
E6D02 00000             CON(5)  =DLITR          DLITERAL
E6D07 00000             CON(5)  =BRNCH          ELSE
E6D0C A6FFF             CON(5)  (INT00)-*
E6D11 00000 INT3        CON(5)  =FLITR          FLITERAL  compile f.p. if compil
E6D16 00000             CON(5)  =DROP           throw away excess copy of DPL
E6D1B 00000             CON(5)  =BRNCH          GO AGAIN
```

```
          0
213 E6D20 65FF          CON(5)  (INT00)-*
          F
214 E6D25 0000 INT4     CON(5)   =DROP          DROP
          0
215 E6D2A 0000          CON(5)   =LITR          LITERAL
          0
216 E6D2F 0000 INT5     CON(5)   =BRNCH         THEN THEN AGAIN
          0
217 E6D34 24FF          CON(5)  (INT00)-*
          F
```



```
218             EJECT
219             ********************************************************************
220             *
221             * -TRAILING  (addr n1 --- addr n2) Adjust char cnt, n1, of text s
222             *       beginning at addr to exclude trailing blanks.
223             *
224             ********************************************************************
225
226 E6D39 F3C6          CON(5)  =!INTRP
          E
227 E6D3E 98   =!TRAIL CON(2)  #89             -TRAILING
228 E6D40 D245        NIBASC  \-TRAILIN\
          2514
          94C4
          94E4
229 E6D50 7C          CON(2)  \G\+#80
230
231 E6D52 0000 =TRAIL  CON(5)  =DOCOL          :
          0
232 E6D57 0000         CON(5)   =DUP           DUP     (addr cnt cnt)
          0
233 E6D5C 0000         CON(5)   =ZERO          0       (addr cnt cnt 0)
          0
234 E6D61 0000         CON(5)   =XDO           DO
          0
235 E6D66 0000 TRL00   CON(5)   =2DUP          2DUP    (addr cnt addr cnt)
          0
236 E6D6B 0000         CON(5)   =ONE-          1-      (addr cnt addr cnt-1)
          0
237 E6D70 0000         CON(5)   =TWO*          2*      (addr cnt addr 2*[cnt-1])
          0
238 E6D75 0000         CON(5)   =ADD           +       (addr cnt addr-of-last-ch
          0
239 E6D7A 0000         CON(5)   =CAT           C@      (addr cnt last-char)
          0
240 E6D7F 0000         CON(5)   =BL            BL      (addr cnt last-char  spac
          0
241 E6D84 0000         CON(5)   =MINUS         -       if last char=space do ELS
          0
242 E6D89 0000         CON(5)   =ZBRNH         IF      otherwise char#space so
          0
243 E6D8E 9100         CON(5)  (TRL0)-*                leave loop!
          0
244 E6D93 0000         CON(5)  =R>             RECOVER LOOP INDEX AND LIMIT
          0
245 E6D98 0000         CON(5)  =R>             AND DROP THEM
          0
246 E6D9D 0000         CON(5)  =2DROP
          0
247 E6DA2 0000         CON(5)  =SEMI           CHEAT AND END HERE
          0
248
249 E6DA7 0000 TRL0    CON(5)   =ONE-          1-      (addr cnt-1)  dropped tra
          0
250 E6DAC 0000 TRL1    CON(5)   =XLOOP         THEN  LOOP      space
```

```
           0
251 E6DB1 5BFF          CON(5)   (TRL00)-*
           F
252 E6DB6 0000          CON(5)   =SEMI          ;
           0
```



```
253                     EJECT
254                *****************************************************************
255                *
256                * IMMEDIATE ( --- ) Mark most recently made dictionary entry
257                *           as a word which will be executed when encountered
258                *           during compilation rather than compiled.
259                *
260                *****************************************************************
261 E6DBB E3D6           CON(5)   !TRAIL
           E
262 E6DC0 98    =!IMMED  CON(2)   #89              IMMEDIATE
263 E6DC2 94D4           NIBASC   \IMMEDIAT\
           D454
           4494
           1445
264 E6DD2 5C             CON(2)   \E\+#80
265
266 E6DD4 0000  =IMMED   CON(5)   =DOCOL           :
           0
267 E6DD9 0000           CON(5)    =LATE           LATEST
           0
268 E6DDE 0000           CON(5)    =LIT
           0
269 E6DE3 0400           CON(5)   #40              40 HEX
           0
270 E6DE8 0000           CON(5)    =TOGG           TOGGLE
           0
271 E6DED 0000           CON(5)    =SEMI           ;
           0
```

```
                    EJECT

                ************************************************************
                *
                * VOCABULARY   ( - )   used as   VOCABULARY FRENCH IMMEDIATE
                *
                * Creates a dictionary entry with normal header followed
                * by another header for the word " " (space), a link to the
                * last word in the new vocabulary, which is initialized to
                * the nfa of the word (e.g.FRENCH) itself, and a link to the
                * previously created vocabulary.
                *
                ************************************************************

E6DF2 0000          CON(5) 0              NO MORE 10 CHAR WORDS
          0
E6DF7 A8    =!VOCAB CON(2) #8A
E6DF9 65F4          NIBASC \VOCABULA\
          3414
          2455
          C414
E6E09 25            NIBASC \R\
E6E0B 9D            CON(2) \Y\+#80

E6E0D 0000 =VOCAB   CON(5) =DOCOL
          0
E6E12 0000          CON(5)  =CREAT        CREATE
          0
E6E17 0000          CON(5)  =LIT
          0
E6E1C 1800          CON(5) #81            81
          0
E6E21 0000          CON(5)  =C,           C,
          0
E6E26 0000          CON(5)  =LIT
          0
E6E2B 0A00          CON(5) #A0            A0
          0
E6E30 0000          CON(5)  =C,           C,
          0
E6E35 0000          CON(5)  =CURR         CURRENT
          0
E6E3A 0000          CON(5)  =AT           @
          0
E6E3F 0000          CON(5)  =AT           get nfa of this word
          0
E6E44 0000          CON(5)  =COMMA        store as last-word link
          0
E6E49 0000          CON(5)  =HERE         HERE
          0
E6E4E 0000          CON(5)  =VOC-L        VOC-LINK
          0
E6E53 0000          CON(5)  =AT           ptr into most recent vocabulary
          0
E6E58 0000          CON(5)  =COMMA        store as parent link
```



```
          0
E6E5D 0000          CON(5)  =VOC-L        VOC-LINK
          0
E6E62 0000          CON(5)  =STORE        store HERE as new most recent vocab
          0
E6E67 0000          CON(5)  =ROM;C        ROM;CODE
          0
E6E6C 0000          CON(5) =DOVOC         dovocabulary
          0
```

```
                        EJECT
                *****************************************************************
                *
                * DEFINITIONS   set CURRENT to the CONTEXT vocabulary, subsequent
                *               definitions will be created in the vocab selected
                *               as the CONTEXT
                *****************************************************************

E6E71 0000          CON(5)  0               NO MORE 11 CHAR WORDS
      0
E6E76 B8    =!DEFIN CON(2)  #8B
E6E78 4454          NIBASC  \DEFINITI\
      6494
      E494
      4594
E6E88 F4E4          NIBASC  \ON\
E6E8C 3D            CON(2)  \S\+#80

E6E8E 0000  =DEFIN  CON(5)  =DOCOL          :
      0
E6E93 0000          CON(5)   =CONTX         CONTEXT
      0
E6E98 0000          CON(5)   =AT            @
      0
E6E9D 0000          CON(5)   =CURR          CURRENT
      0
E6EA2 0000          CON(5)   =STORE         !
      0
E6EA7 0000          CON(5)   =SEMI          ;
      0
```



```
                        EJECT

                ****************************************
                *
                * STRING-ARRAY
                * Use is: MAX DIM STRING-ARRAY <name>
                * String-array has form max dim (1 byte each),
                * followed by max*(dim+2) bytes.
                *
                ************************************************

E6EAC 0000          CON(5)  0               NO MORE 12 CHAR WORDS
      0
E6EB1 C8    =!STRAR CON(2)  #8C
E6EB3 3545          NIBASC  \STRING-\
      2594
      E474
      D2
E6EC1 1425          NIBASC  \ARRA\
      2514
E6EC9 9D            CON(2) \Y\+#80
E6ECB 0000  =STRAR  CON(5)   =DOCOL         :
      0
E6ED0 0000          CON(5)   =CREAT         CREATE
      0
E6ED5 0000          CON(5)   =ROMC*         ROM;CODE with DOSTRA
      0
                * max dim
E6EDA 0000          CON(5)   =M255          REPLACE DIM WITH 255 IF >255
      0
E6EDF 0000          CON(5)   =SWAP          SWAP
      0
E6EE4 0000          CON(5)   =M255          REPLACE MAX "    "   "  "
      0
E6EE9 0000          CON(5)   =2DUP          2DUP
      0
E6EEE 0000          CON(5)   =HERE          HERE
      0
                * dim max dim max here
E6EF3 0000          CON(5)   =C!            C!
      0
E6EF8 0000          CON(5)   =HERE          HERE
      0
E6EFD 0000          CON(5)   =TWO+          2+
      0
E6F02 0000          CON(5)   =C!            C!
      0
                * dim max
E6F07 0000          CON(5)   =2DUP          2DUP
      0
E6F0C 0000          CON(5)   =HERE          HERE
      0
E6F11 0000          CON(5)   =FOUR+         4+
      0
E6F16 0000          CON(5)   =ROT           ROT
      0
```

```
366 E6F1B 0000        CON(5)   =ROT        ROT
          0
367             * dim max here+ dim max
368 E6F20 0000        CON(5)   =TWO+       2+
          0
369 E6F25 0000        CON(5)   =MULT       *
          0
370 E6F2A 0000        CON(5)   =TWO*       2*
          0
371 E6F2F 0000        CON(5)   =FOUR+      4+
          0
372 E6F34 0000        CON(5)   =ALLOT      NALLOT
          0
373             * NOW GO THROUGH PUTTING IN MAX 0
374             * AT BEGINNING OF EACH ELEMENT
375             *
376             * dim max here+  (counter max ptr)
377             *                          BEGIN
378 E6F39 0000        CON(5)   =ROT      max ptr counter
          0
379 E6F3E 0000        CON(5)   =>R       max ptr    R:counter
          0
380 E6F43 0000 STRA1  CON(5)   =R>        max ptr counter
          0
381 E6F48 0000        CON(5)   =?DUP
          0
382 E6F4D 0000        CON(5)   =ZBRNH      WHILE
          0
383 E6F52 0500        CON(5)  (STRA2)-*
          0
384 E6F57 0000        CON(5)   =ONE-       decrement counter
          0
385 E6F5C 0000        CON(5)   =>R         max ptr  R:counter
          0
386 E6F61 0000        CON(5)   =2DUP       max ptr max ptr
          0
387 E6F66 0000        CON(5)   =C!         C!
          0
388 E6F6B 0000        CON(5)   =DUP        max ptr ptr
          0
389 E6F70 0000        CON(5)   =TWO+       max ptr ptr+
          0
390 E6F75 0000        CON(5)   =ZERO       0
          0
391 E6F7A 0000        CON(5)   =SWAP       max ptr 0 ptr+
          0
392 E6F7F 0000        CON(5)   =C!         C!
          0
393 E6F84 0000        CON(5)   =FOUR+      max ptr+4
          0
394             * max ptr(past str len bytes)
395 E6F89 0000        CON(5)   =OVER       max ptr max
          0
396 E6F8E 0000        CON(5)   =TWO*       max ptr 2*max
          0
```



```
397 E6F93 0000        CON(5)   =ADD        max newptr
          0
398 E6F98 0000        CON(5)   =BRNCH      REPEAT
          0
399 E6F9D 6AFF        CON(5)  (STRA1)-*
          F
400 E6FA2 0000 STRA2  CON(5)   =2DROP      2DROP
          0
401 E6FA7 0000        CON(5)   =SEMI       ;
          0
402
403 E6FAC 1BF6 =ERR?  CON(5)   =$ERR?
          E
404 E6FB1 8E00 =$ERR? GOSUBL   =DOUSE
          00
405 E6FB7 BBBF        CON(5)   =oERR?
          2
406
407
408 E6FBC       =BEND6  END
```

```
=!?TERM  Abs  944901 #E6B05 -   20   48
=!DEFIN  Abs  945782 #E6E76 -  321
=!FCON   Abs  944966 #E6B46 -   49   71
=!FVAR   Abs  945036 #E6B8C -   72  112
=!IMMED  Abs  945600 #E6DC0 -  262
=!INTRP  Abs  945215 #E6C3F -  160  226
=!SECD   Abs  945107 #E6BD3 -  113  131
=!STRAR  Abs  945841 #E6EB1 -  343
=!TRAIL  Abs  945470 #E6D3E -  227  261
=!VOCAB  Abs  945655 #E6DF7 -  287
=![CMP]  Abs  945148 #E6BFC -  132  159
=$?TERM  Abs  944926 #E6B1E -   24   23
=$ERR?   Abs  946097 #E6FB1 -  404  403
=$HPLP   Abs  945091 #E6BC3 -  100   98
=$SECD   Abs  945132 #E6BEC -  118  116
 -FIND   Ext          -      138  175
 2DROP   Ext          -      246  400
 2DUP    Ext          -      235  354  362  386
 >R      Ext          -      379  385
 ?COMP   Ext          -      137
 ?DUP    Ext          -      381
=?TERM   Abs  944921 #E6B19 -   23
 ABORTW  Ext          -      142
 ADD     Ext          -      238  397
 ALLOT   Ext          -       59   80  372
 AT      Ext          -      167  179  196  301  302  306  328
 BL      Ext          -      240
 BRNCH   Ext          -      184  188  208  212  216  398
 C!      Ext          -      357  360  387  392
 C,      Ext          -      296  299
 CAT     Ext          -      239
 COMMA   Ext          -      145  183  303  307
 CONTX   Ext          -      327
 CREAT   Ext          -       54   77  293  348
 CURR    Ext          -      300  329
=DEFIN   Abs  945806 #E6E8E -  326
 DLITR   Ext          -      207
 DOCOL   Ext          -       53   76  136  164  174  231  266  292
                             326  347
 DOFCON  Ext          -       61
 DOUSE   Ext          -      100  118  404
 DOVOC   Ext          -      311
 DPL     Ext          -      195
 DROP    Ext          -      144  211  214
 DUP     Ext          -      197  232  388
 EQUAL   Ext          -      200
=ERR?    Abs  946092 #E6FAC -  403
 EXEC    Ext          -      168  186
=FCON    Abs  944986 #E6B5A -   53
 FLITR   Ext          -      210
 FOUR+   Ext          -      364  371  393
=FVAR    Abs  945056 #E6BA0 -   76
 HERE    Ext          -       55  191  304  355  358  363
=HPLP    Abs  945086 #E6BBE -   98
=IINTRP  Abs  945265 #E6C71 -  174
```



```
=IMMED   Abs  945620 #E6DD4 -  266
 INT0    Abs  945340 #E6CBC -  191  177
 INT00   Abs  945270 #E6C76 -  175  185  189  209  213  217
 INT1    Abs  945325 #E6CAD -  186  182
 INT2    Abs  945330 #E6CB2 -  188
 INT3    Abs  945425 #E6D11 -  210  203
 INT4    Abs  945445 #E6D25 -  214  206
 INT5    Abs  945455 #E6D2F -  216
=INTRP   Abs  945235 #E6C53 -  164
 KEYPTR  Ext          -       25
 LATE    Ext          -      267
 LIT     Ext          -       57   78  165  198  268  294  297
 LITR    Ext          -      215
 LT      Ext          -      180
 M255    Ext          -      351  353
 MINUS   Ext          -      241
 MULT    Ext          -      369
 NUMB    Ext          -      192
 ONE+    Ext          -      204
 ONE-    Ext          -      236  249  384
 OVER    Ext          -      395
 R>      Ext          -      244  245  380
=REND6   Abs  946108 #E6FBC -  408
 ROM;C   Ext          -       60  310
 ROMC*   Ext          -      349
 ROT     Ext          -      365  366  378
=SECD    Abs  945127 #E6BE7 -  116
 SEMI    Ext          -       81  146  169  247  252  271  331  401
 STATE   Ext          -      178
 STO     Ext          -       56
 STORE   Ext          -      309  330
 STRA1   Abs  945987 #E6F43 -  380  399
 STRA2   Abs  946082 #E6FA2 -  400  383
=STRAR   Abs  945867 #E6ECB -  347
 SWAP    Ext          -      352  391
 TERMX   Abs  944953 #E6B39 -   32   30
 TOGG    Ext          -      270
=TRAIL   Abs  945490 #E6D52 -  231
 TRL0    Abs  945575 #E6DA7 -  249  243
 TRL00   Abs  945510 #E6D66 -  235  251
 TRL1    Abs  945580 #E6DAC -  250
 TWO*    Ext          -      237  370  396
 TWO+    Ext          -      359  368  389
 VOC-L   Ext          -      305  308
=VOCAB   Abs  945677 #E6E0D -  292
 XDO     Ext          -      234
 XLOOP   Ext          -      250
 ZBRNH   Ext          -      176  181  202  205  242  382
 ZEQ     Ext          -      141  201
 ZERO    Ext          -      233  390
=[CMP]   Abs  945168 #E6C10 -  136
```

Input Parameters

Source file name is MR&FT6

Listing file name is MR/FT6::65

Object file name is MR%FT6::65

```
                                111111
                       0123456789012345
```

Initial flag settings are

Errors

None

Saturn Assembler News

```
                        TITLE  FT7:_PROLOGUES
                *       RDSYMB MR%FT6
                *       ABS    REND6
                        RDSYMB MR%GTO
                        RDSYMB MR%TIQ
E7000                   ABS    F7STRT
                ********************************************
                *                                          *
                *  MR&FT7      <840608.1403>                *
                *                                          *
                * Prologues, etc.                          *
                *                                          *
                ********************************************
                *TYTABF = F7STRT IN MR%GTO
                *
                * FILE TYPE TABLE
                *
                * Used to supply file type since type FORTH is
                * not known to mainframe. See CATPOL in MR/FTO.
                *
E7000 0                 CON(1)  0               mainframe format
E7001 0                 CON(1)  0               mainframe copy
E7002 0                 CON(1)  0               non-execute
E7003 50                CON(2)  5               no sub-header
E7005 64F4              NIBASC  \FORTH\
      2545
      84
E700F 2                 CON(1)  2               2 types
E7010 812E              CON(4)  =FthFil          ordinary FORTH
E7014 912E              CON(4)  (=FthFil)+1      with SECURE
E7018 FF                NIBHEX  FF              end of table
                ************************************
                *
                * DOCOLON
                * on entry, A[A]=WA, unincremented
                *           D0=I, incremented
                *
                * A-register points to the DOCOL in secondary,
                * since we come here right from NEXT. We
                * will increment it and make it the new instruction
                * pointer, after saving the old instruction pointer (in
                * D0, of course) on the return stack (pointed to by the
                * B-register).
                *
                ************************************

E701A D9        =DOCOL  C=B     A               get rtn stk ptr
E701C 136               CD0EX                   D0=rtn stk;c= instr ptr
E701F 184               D0=D0-  5               push item on RS
E7022 144               DAT0=C  A               old I-->RS
E7025 132               AD0EX
E7028 D8                B=A     A               put RS back to B[A]
E702A 164               D0=D0+  5               old WA+5-->I
E702D 03                RTNCC
```



```
                ************************************
                *
                * DOCONSTANT
                *
                * The runtime code for a constant.
                * A-register points to the cfa, where (=DOCON) is,
                * so we just increment it (pointing to pfa) and get
                * the value there; then push it on stack.
                *
                ************************************

E702F 132       =DOCON  AD0EX                   WA-->D0
E7032 164               D0=D0+  5               inc WA
E7035 146               C=DAT0  A               get the value of the constant
E7038 130               D0=A                    restore I
E703B 1C4               D1=D1-  5
E703E 145               DAT1=C  A               push value onto SP
E7041 03                RTNCC

                ************************
                *
                * DOFCONSTANT  (FLOATING POINT CONSTANT
                *
                * Same idea as with DOCON, but instead of pushing
                * the value (here 16 nibs instead of 5) stored at pfa,
                * we have to do a stack-lift on X,Y,Z,T locations and
                * write the value to the X-register location.
                *
                ************************

E7043 132       =DOFCON AD0EX
E7046 164               D0=D0+ 5
E7049 1567              C=DAT0 W                READ F.P. VALUE
E704D 130               D0=A                    RECOVER I
E7050 10B               R3=C                    SAVE CONSTANT
E7053 7321              GOSUB  SAVEFP           SAVE FORTH POINTERS
E7057 75C2              GOSUB  =STKLFT          LIFT STACK
E705B 1B0D              D0=(5)  =oX
      BF2
E7062 113               A=R3
E7065 1507              DAT0=A W                WRITE CONSTANT TO X
E7069 6B31              GOTO   GETFP            RECOVER FORTH POINTERS

                ************************
                *
                * DOVARIABLE
                *
                * Runtime code for a variable.
                * As in above routines, A-register points to
                * cfa on entry; what is supposed to be returned is
                * the pfa, which is [A]+5.
                *
                ************************

E706D 20        =DOVAR  P=      0               SET FOR C LOAD
```

```
108 E706F 3450          LC(5)   5
          000
109 E7076 CA    dv      A=A+C   A               inc WA to get variable address
110 E7078 1C4           D1=D1-  5
111 E707B 141           DAT1=A  A               push addr onto SP
112 E707E 03            RTNCC
113
114
115             *******************************
116             *
117             * DOSTRING
118             *
119             * Instead of returning pfa(=cfa+5), a string
120             * variable wants to return: pfa+4, which is the
121             * address of the first character in the string
122             * (after the maxlen&curlen bytes); and then the
123             * current length on top of that.
124             *
125             *******************************
126
127 E7080 20    =DOSTR P=       0
128 E7082 3490          LC(5)   9               point to pfa+4
          000
129 E7089 79EF          GOSUB   dv              push pfa+4 on stack
130 E708D 132           ADOEX                   D0-->pfa+4
131 E7090 181           D0=D0-  2               point to current length
132 E7093 14E           C=DAT0  B               get it
133 E7096 161           D0=D0+  2
134 E7099 132           ADOEX                   restore A & D0
135 E709C 1C4           D1=D1-  5
136 E709F 145           DAT1=C  A               push curlen
137 E70A2 03            RTNCC
138
139             *************************
140             *
141             * DOSTRINGARRAY
142             *
143             * Get index off stack; check that there is
144             * one and it is valid; point to the indexed
145             * array element and return its length
146             * and address.
147             *
148             ***********************************************
149 E70A4 31E2 DOSTer   LC(2)   =eBADPM
150 E70A8 22            P=      2
151 E70AA 31F2          LC(2)   =ID
152 E70AE 20            P=      0
153 E70B0 8D00          GOVLNG  =BSERR
          000
154             * * * * * * * * * * * * * * * * * * * * * * * *
155             *
156 E70B7 20    =DOSTRA P=      0
157 E70B9 136           CD0EX
158 E70BC 108           R0=C                    SAVE D0
159 E70BF D9            C=B     A
```



```
160 E70C1 109           R1=C                    SAVE B
161 E70C4 AF2           C=0     W
162 E70C7 147           C=DAT1  A               GET INDEX OF ARRAY
163 E70CA 8AA           ?C=0    A               ZERO ALWAYS BAD
164 E70CD 7D            GOYES   DOSTer
165 E70CF D7            D=C     A               D=index of array
166             * A-register points to cfa after NEXT
167 E70D1 130           D0=A                    D0-->CFA
168 E70D4 166           D0=D0+  7               POINT TO PFA+2=DIMENSION
169 E70D7 D2          C=0     A
170 E70D9 14E           C=DAT0  B               get DIMENSION of array
171             * is dimension less than the index given to us?
172 E70DC 8B3           ?C<D    A               index out of range?
173 E70DF 5C            GOYES   DOSTer
174 E70E1 181            D0=D0-  2             POINT BACK TO MAXLEN
175 E70E4 174           D1=D1+  5               UPDATE STACK PTR
176 E70E7 14E           C=DAT0  B               GET MAXLEN
177 E70EA E6            C=C+1   A
178 E70EC E6            C=C+1   A               add 2 for max/cur bytes
179 E70EE C6            C=C+C   A               bytes to nibs
180 E70F0 D5            B=C     A               B=len of each string
181 E70F2 163           D0=D0+  4               point to 1st string
182 E70F5 136           CD0EX                   get ptr into array
183 E70F8 CF    NSTR    D=D-1   A               count down index
184 E70FA 8AB           ?D=0    A
185 E70FD 80            GOYES   DSTR
186 E70FF C9            C=C+B   A               add string to ptr
187 E7101 66FF          GOTO    NSTR
188 E7105 137   DSTR    CD1EX                   D1-->STR
189 E7108 10A           R2=C                    save data stack pointer
190 E710B 171           D1=D1+  2
191 E710E 14F           C=DAT1  B               get len
192 E7111 AE7           D=C     B               D=lenght of string
193 E7114 171           D1=D1+  2               beginning of str
194 E7117 11A           C=R2                    retrieve stk ptr
195 E711A 137           CD1EX                   D1=STK,C=str addr
196 E711D 1C4           D1=D1-  5               push item
197 E7120 145           DAT1=C  A                    addr
198 E7123 1C4           D1=D1-  5               push another
199 E7126 D2            C=0     A
200 E7128 AEB           C=D     B               get len
201 E712B 145           DAT1=C  A                    len on stack
202 E712E 119           C=R1                    get back B
203 E7131 D5            B=C     A
204 E7133 118           C=R0                    get back D0
205 E7136 136           CD0EX
206 E7139 03            RTNCC
207
208             ************************************************************
209             *
210             * DODOES
211             *
212             * Subroutine of secondary DOES (in MR/FT3) and DOVOC below.
213             * Like DOCOL plus it returns pfa onto stack.
214             * on entry, RSTK  points to the list of cfas following DOES>
```

```
                *                                   in the defining word
                *           A[A]+5 points to the parameter field of the word
                *                                   being executed
                * the action of dodoes is:
                *       1. push I onto RS
                *       2. set new I to the cfa list following DOES>
                *       3. push the pfa of the word being executed onto SP
                *
                * To see why we want to do this, look at DOVOC below.
                *
                ************************************************************

E713B D9        =DODOES C=B     A
E713D 136               CD0EX
E7140 184               D0=D0-  5
E7143 144               DAT0=C  A               push I onto RS
E7146 07                C=RSTK
E7148 136               CD0EX                   put up new I
E714B D5                B=C     A               put RS ptr back to B.A
E714D 20                P=      0
E714F 3450              LC(5)   5
      000
E7156 CA                A=A+C   A               increment WA to get pfa
E7158 1C4               D1=D1-  5
E715B 141               DAT1=A  A               pfa to SP
E715E 03                RTNCC

                ************************************************************
                *
                * DOVOCABULARY
                *
                * Get pfa of vocabulary. At pfa is:
                *                           * 180A  (dummy header)
                *                           * last-word field
                *                           * parent vocab field
                *
                * Skip the dummy header, store addr of last-word link
                * in CONTEXT.
                *
                ************************************************************

E7160 8E5D      =DOVOC  GOSUBL  DODOES
      FF
E7166 0000              CON(5)  =FOUR+      skip dummy header
      0
E716B 0000              CON(5)  =CONTX
      0
E7170 0000              CON(5)  =STORE
      0
E7175 8E17              CON(5)  =SEMI
      E
```



```
                        EJECT
                ************************************
                * SAVEFP  --  SAVE FORTH POINTERS
                *
                * IN: D1= DATA STK PT
                *     D0= I
                *     B(A)=RTN STK PTR
                *
                * OUT: SAVES D0, D1, B(A) IN RAM
                *      CARRY CLEAR
                *
                * USES: C(A), D0, P
                *
                ************************************
E717A 20        =SAVEFP P=      0
E717C 34DF      LC(5)   =DSTKAD         C(A)=DATA STK SAVE AREA
      AF2
E7183 137               CD1EX                   ADDR TO D1, CURRENT DATA STK TO C
E7186 145               DAT1=C A                SAVE DATA STK PT
E7189 174               D1=D1+ 5
E718C D9                C=B    A
E718E 145               DAT1=C A                SAVE RTN STK PT
E7191 174               D1=D1+ 5
E7194 136               CD0EX
E7197 145               DAT1=C A                SAVE I
E719A 1C9               D1=D1- 10               POINT TO ORIGINAL D1
E719D 147               C=DAT1 A
E71A0 137               CD1EX
E71A3 03                RTNCC


                ********************************************************************
                * GETFP  --  GET FORTH POINTERS
                *
                * IN: DOESN'T MATTER
                *
                * ASSUME: P= 0, HEXMODE
                *
                * OUT: D1= DATA STK PT
                *      D0= I
                *      B(A)= RTN STK PT
                *      CARRY CLEAR
                *
                * USES: C(A), B, D0, D1, P
                *

E71A5 20        =GETFP  P=      0
E71A7 3470      LC(5)   =IREG           START LOADING I
      BF2
E71AE 137               CD1EX                   SAVE ADDRESSES TO D1
E71B1 147               C=DAT1 A
E71B4 134               D0=C                    D0= I
E71B7 1C4               D1=D1- 5
E71BA 147               C=DAT1 A
E71BD D5                B=C    A                B= RTN STK PT
```

```
314 E71BF 1C4          D1=D1- 5
315 E71C2 147          C=DAT1 A
316 E71C5 135          D1=C                    D1= DATA STK
317 E71C8 03           RTNCC
318
```



```
319                    EJECT
320             ************************************************************
321             *                                                          *
322             * GENERAL USE OF CPU REGISTERS IN FORTH SYSTEM             *
323             *                                                          *
324             * B(A)=  RTN STACK POINTER  (points at top item --         *
325             *                            decreasing stack)
326             * D1=    FORTH DATA STACK POINTER   (points at top item -- *
327             *                                    decreasing stack)
328             * D0=    I  (instruction pointer)                          *
329             * A(A)=  WA (word address)                                 *
330             *                                                          *
331             ************************************************************
332
333
334
335             ***************************************
336             * INNER LOOP
337             *
338             * NEXT  @I--->WA
339             *       I=I+5          TYPICAL # STATES: 98
340             *       @WA-->PC
341             *
342             ***************************************
343
344 E71CA 142  =NEXT   A=DAT0  A       A(A)=WA  [18]
345 E71CD 164         D0=D0+  5       incr I to next addr [7]
346 E71D0 132  =RUN    AD0EX           D0=WA, A=I  [8]
347 E71D3 146         C=DAT0  A       C=@WA       [18]
348 E71D6 132         AD0EX           D0=I, A(A)=WA   [8]
349 E71D9 06          RSTK=C          push @WA to PC  [8]
350 E71DB 01          RTN                             [9]
351
352 E71DD 79EF =NEXT00 GOSUB   NEXT    put NEXT01's addr on CPU RTN STK
353             *                           [12]
354 E71E1 5BF  NEXT01  GONC    NEXT00  NOTE: we're counting on NO CARRY!!!
355             *                           [10]
356 E71E4 68FF         GOTO    NEXT00   This is insurance
```

```
                        EJECT
                *
                * SEMI  POP RTN STK--->I
                *       GOTO NEXT
                *
                * This routine is also called EXIT
                *
E71E8           =EXIT
E71E8 DE17E     =SEMI   CON(5)  =$SEMI
E71ED 7310      =$SEMI  GOSUB   DOATN
E71F1 D9                C=B     A       RTN STK PTR to C
E71F3 136               CD0EX           D0=RTN STK, C=I
E71F6 142               A=DAT0  A       A= new I
E71F9 164               D0=D0+  5       decrement RTN STK PTR
E71FC 132               AD0EX           D0=new I, A=RTN STK PTR
E71FF D8                B=A     A       restore RTN STK PTR
E7201 58C               GONC    NEXT


                ********************************************************
                *
                * This rtne checks to see if the ATTN key was hit
                * and if it was it runs the ABORT routine.
                * It also handles other service requests.  It is
                * run after every BRANCH and OBRANCH and SEMI
                ********************************************************

E7204 727F      =DOATN  GOSUB   =SAVEFP      SAVE FORTH POINTERS
E7208 8F000     000     GOSBVL  =CK"ON"      CKECK IF ATTN KEY HIT
E720F 5E2               GONC    ATTN         WAS HIT
E7212 87C               ?ST=1   Except
E7215 D0                GOYES   EXCP10
E7217 80E               SREQ?                SERV. REQ?
E721A 834               ?SR=0
E721D D1                GOYES   EXCP99       EXIT
E721F 824               SR=0
E7222 8F000     EXCP10  GOSBVL  =CKSREQ      CHECK SERVICE REQUEST
                000
E7229 86C               ?ST=0   Except       CLEARED?
E722C E0                GOYES   EXCP99       EXIT
E722E 84C               ST=0    Except       CLEAR IT
E7231 8F000             GOSBVL   =FPOLL
                000
E7238 8F                CON(2)  =pExcpt
E723A 6A6F      EXCP99  GOTO    =GETFP       RESTORE FORTH POINTERS


E723E 84C       ATTN    ST=0    Except       clear exception flag
E7241 1F000             D1=(5)  =ATNFLG
                000
```



```
E7248 D0                A=0     A
E724A 1590              DAT1=A  1            CLEAR ATTN FLAG
E724E 735F              GOSUB   GETFP
E7252 07                C=RSTK               THROW AWAY 1 RTN ADDR
E7254 7A00              GOSUB   SETI0
E7258 000C              CON(5)  =CR          DO A CARRIAGE RETURN
      2
E725D 0000              CON(5)  =ABORT       NOW DO ABORT
      0

E7262 07        SETI0   C=RSTK               POP ADDR
E7264 136               CD0EX                D0=NEW I
E7267 626F              GOTO    NEXT         GO TO INNER LOOP
```

```
                        EJECT
                *************************************************
                *
                * THIS ROUTINE ALLOWS FOR RETURN TO BASIC KEYWORDS
                * ASSUMES THE RTN ADDR IS ON STK
                *
                *************************************************

E726B 0727E     =BSCRTN CON(5) =$BSRTN
E7270 147       =$BSRTN C=DAT1 A
E7273 174               D1=D1+ 5
E7276 1BDFAF2           D0=(5)  =DSTKAD
E727D 137               CD1EX
E7280 144               DAT0=C  A               SAVE TOS
E7283 137               CD1EX
E7286 06                RSTK=C
E7288 01                RTN
```



```
                        STITLE FLOATING POINT SUBROUTINES
                *************************************************
                *
                * GETXLN
                *
                * GET X & LASTX AS 15-FORM NUMBERS IN
                * REGISTERS (C,D) AND (A,B)
                *
                *************************************************

E728A 1B0DBF2   =GETX   D0=(5)  (=oX)
E7291 1527      GX00    A=DAT0  W               READ CONTENTS OF X REGISTER
E7295 05        GX01    SETDEC
E7297 1F000     000     D1=(5) =MTHSTK          D1 MUST POINT TO MATHSTK
E729E 147               C=DAT1 A                FOR WARNINGS, ERRORS
E72A1 135               D1=C
E72A4 8F000     000     GOSBVL =SIGTST          TEST FOR NaNs, INFs
E72AB 5C0               GONC   GX02
E72AE 8F000     000     GOSBVL =uRES12
E72B5 AFA               A=C    W
E72B8 8D000     000 GX02    GOVLNG  =SPLITA     GET # IN (A,B)

E72BF 77CF      =GETXLN GOSUB   GETX            this used to be Y
                *                               before stkdrp
E72C3 8F000     000     GOSBVL =STAB1           A,B stored in R0,R1
E72CA 18F               D0=D0-  16              POINT TO LASTX
E72CD 70CF              GOSUB   GX00            get X from LASTX
E72D1 8F000     000     GOSBVL =RCCD1           Y from R0,R1
E72D8 8D000     000     GOVLNG =XYEX            exchange X,Y

                *
                * GETX+L
                *
                * Gets X in (A,B); also puts X in LASTX
                *
E72DF 1B0DBF2   =GETX+L D0=(5)  (=oX)
E72E6 1527              A=DAT0  W
E72EA 18F               D0=D0-  16              LASTX
E72ED 1507              DAT0=A  W               LASTX:=X
E72F1 63AF              GOTO    GX01
                *
                * PUTABX
                *
                * PUT A RESULT OF COMPUTATION IN (A,B) INTO
                * VARIABLE (41-C REGISTER) X
                *
```

```
481 E72F5 1F00 =PUTABX D1=(5)  =MTHSTK
          000
482 E72FC AFF          CDEX    W
483 E72FF 147          C=DAT1  A
484 E7302 135          D1=C
485 E7305 AFF          CDEX    W
486 E7308 8F00         GOSBVL  =uRES12     PACK UP # INTO C
          000
487 E730F 1B0D         D0=(5)  (=oX)
          BF2
488 E7316 1547         DAT0=C  W
489 E731A 04           SETHEX
490 E731C 688E         GOTO    GETFP       RECOVER FORTH POINTERS
491
492           *
493           * STACK_LIFT
494           *                 ---> LOST
495           *    +---------+  |           +----------+
496           *    |   T     |--  ---------->|     T    |
497           *    +---------+    |          +----------+
498           *    |   Z     |-----  ------->|     Z    |
499           *    +---------+       |      +----------+
500           *    |   Y     |--------  ---->|     Y    |
501           *    +---------+          |    +----------+
502           *    |   X     |-----------    |     X    |
503           *    +---------+               +----------+
504           *    |   L     |               |     L    |
505           *    +---------+               +----------+
506           *
507
508 E7320 20   =STKLFT P=       0
509 E7322 1F0D         D1=(5)  =oX           POINT TO X
          BF2
510 E7329 1577         C=DAT1  W             READ X
511 E732D 17F          D1=D1+  16            POINT TO Y
512 E7330 1537         A=DAT1  W             READ Y
513 E7334 1557         DAT1=C  W             WRITE X TO Y
514 E7338 17F          D1=D1+  16            POINT TO Z
515 E733B 1577         C=DAT1  W             READ Z
516 E733F 1517         DAT1=A  W             WRITE Y TO Z
517 E7343 17F          D1=D1+  16            POINT TO T
518 E7346 1557         DAT1=C  W             WRITE Z TO T
519 E734A 01           RTN
520
521
522           *
523           * STACK_DROP
524           *
525           *    +---------+               +----------+
526           *    |   T     |----------     |     T    |
527           *    +---------+          |    +----------+
528           *    |   Z     |--------  -----|     Z    |
529           *    +---------+        |      +----------+
530           *    |   Y     |------  ------->|     Y    |
531           *    +---------+      |        +----------+
```



```
532           *    |   X     |---  --------->|     X    |
533           *    +---------+  |            +----------+
534           *    |   L     |  ------------>|     L    |
535           *    +---------+               +----------+
536           *
537           *
538
539 E734C 20   =STKDRP P=       0
540 E734E 1F00         D1=(5)  =oT           POINT TO T
          CF2
541 E7355 1577         C=DAT1  W             READ T
542 E7359 1CF          D1=D1-  16            POINT TO Z
543 E735C 1537         A=DAT1  W             READ Z
544 E7360 1557         DAT1=C  W             WRITE T TO Z
545 E7364 1CF          D1=D1-  16            POINT TO Y
546 E7367 1577         C=DAT1  W             READ Y
547 E736B 1517         DAT1=A  W             WRITE Z TO Y
548 E736F 1CF          D1=D1-  16            POINT TO X
549 E7372 1537         A=DAT1  W             READ X
550 E7376 1557         DAT1=C  W             WRITE Y TO X
551 E737A 1CF          D1=D1-  16            POINT TO LASTX
552 E737D 1517         DAT1=A  W             WRITE X TO LASTX
553 E7381 01           RTN
554
555
556 E7383              END
```

=$BSRTN	Abs	946800	#E7270	-	427	426			
=$SEMI	Abs	946669	#E71ED	-	368	367			
ABORT	Ext			-	413				
ATNFLG	Ext			-	406				
ATTN	Abs	946750	#E723E	-	405	389			
=BSCRTN	Abs	946795	#E726B	-	426				
BSERR	Ext			-	153				
CK"ON"	Ext			-	388				
CKSREQ	Ext			-	396				
CONTX	Ext			-	257				
=DOATN	Abs	946692	#E7204	-	387	368			
=DOCOL	Abs	946202	#E701A	-	45				
=DOCON	Abs	946223	#E702F	-	65				
=DODOES	Abs	946491	#E713B	-	227	255			
=DOFCON	Abs	946243	#E7043	-	84				
=DOSTR	Abs	946304	#E7080	-	127				
=DOSTRA	Abs	946359	#E70B7	-	156				
DOSTer	Abs	946340	#E70A4	-	149	164	173		
=DOVAR	Abs	946285	#E706D	-	107				
=DOVOC	Abs	946528	#E7160	-	255				
DSTR	Abs	946437	#E7105	-	188	185			
EXCP10	Abs	946722	#E7222	-	396	391			
EXCP99	Abs	946746	#E723A	-	402	394	398		
=EXIT	Abs	946664	#E71E8	-	366				
FOUR+	Ext			-	256				
FPOLL	Ext			-	400				
=GETFP	Abs	946597	#E71A5	-	306	94	402	409	490
=GETX	Abs	946826	#E728A	-	445	457			
=GETX+L	Abs	946911	#E72DF	-	470				
=GETXLN	Abs	946879	#E72BF	-	457				
GX00	Abs	946833	#E7291	-	446	461			
GX01	Abs	946837	#E7295	-	447	474			
GX02	Abs	946872	#E72B8	-	455	452			
MTHSTK	Ext			-	448	481			
=NEXT	Abs	946634	#E71CA	-	344	352	375	417	
=NEXT00	Abs	946653	#E71DD	-	352	354	356		
NEXT01	Abs	946657	#E71E1	-	354				
NSTR	Abs	946424	#E70F8	-	183	187			
=PUTABX	Abs	946933	#E72F5	-	481				
RCCD1	Ext			-	462				
=RUN	Abs	946640	#E71D0	-	346				
=SAVEFP	Abs	946554	#E717A	-	275	89	387		
=SEMI	Abs	946664	#E71E8	-	367	259			
SETI0	Abs	946786	#E7262	-	415	411			
SIGTST	Ext			-	451				
SPLITA	Ext			-	455				
STAB1	Ext			-	459				
=STKDRP	Abs	947020	#E734C	-	539				
=STKLFT	Abs	946976	#E7320	-	508	90			
STORE	Ext			-	258				
XYEX	Ext			-	463				
dv	Abs	946294	#E7076	-	109	129			
uRES12	Ext			-	453	486			

Input Parameters

Source file name is MR&FT7

Listing file name is MR/FT7::65

Object file name is MR%FT7::65

```
                                    111111
                          0123456789012345
```

Initial flag settings are

Errors

None

Saturn Assembler News

HP-71 Assember

Internal Design Specification

FORTH/Assembler ROM

2:20 PM THU., 10 MAY , 1984

Overview	CHAPTER 1

The Assembler is a typical two pass assembler. The first pass is spent processing labels & calculating the length of the object file. The second pass is spent calculating the object code, outputting the object file, and creating the listing file.

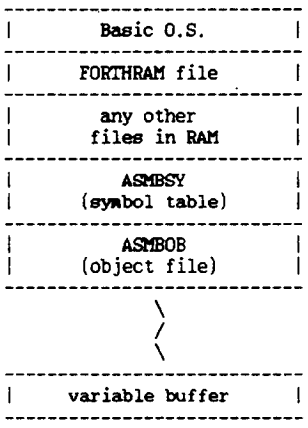
This document describes the Assembler by looking at each of the major parts: variables, symbol table, expressions, and opcode processing. It assumes a knowledge of HP-71 FORTH, how the Assembler is executed, and a rudimentary knowledge of assemblers.

2:20 PM THU., 10 MAY , 1984

Memory Layout During Assembly	CHAPTER 2

The assembler must handle four different data areas during execution. These are: the FORTHRAM file, the Symbol Table (ASMBSY), the Object File (ASMBOB), and the variable buffer. During a typical assembly, the memory will look like:

Memory Requirements



The assembler creates the ASMBOB file, whether it needs it or not. When a LEX or BIN pseudo-op is reached (1st line), it realizes that this file is where the output will go. If a FORTH pseudo-op is reached, it purges this file from the file chain and realizes that FORTHRAM is where the output will go.

2.1 Memory Requirements

The amount of memory required to run the assembler can be broken down into parts. The following diagram shows all possible sections of the assembler which require memory:

Memory Requirements

Symbol Table : 37 + 18 + 18 * <entries in symbol table>

Object File : FORTH: 38
LEX or BIN: <size of object file>

Variables : 1332

Calls to Basic : 160

Listing File : HP-IL: 0
RAM: <dependent on file size. Not predictable>
none: 0

Minimum : 1585 nibbles

Program Flow	CHAPTER 3

The assembler uses a two pass technique. The major tasks of pass one are to process all labels & equates and to accumulate the precise size of the object file in nibbles. The major tasks of pass two are to output the object code, create the listing file, and issue errors.

To illustrate the operation of the assembler, we will take a sample source line and detail which words get executed. We will further specify the words by adding, in parentheses following the word name, the number of the FORTH/Assembler ROM source file that contains the word. The source files are MR&AS0 through MR&AS8. Each source file has a title:

```
MR&AS0: OPCODE LOOKUP TABLES
MR&AS1: VARIABLES
MR&AS2: IO
MR&AS3: SYMBOL TABLE
MR&AS4: EXPRESSIONS
MR&AS5: OPCODES
MR&AS6: PSEUDO-OPS
MR&AS7: PARSING
MR&AS8: MAIN LOOP
```

The notation VAR.INIT(1) will indicate that VAR.INIT is in MR&AS1: Variables. The notation DROP(SYS) will indicate that the word DROP is part of the FORTH system, and not unique to the assembler. The sample source line is:

```
=BREAK  LC(5)  'AB'*16+2
```

In the sequence that follows, words called by other words are listed indented below the calling word.

Pass One Flow:

```
DO.LINE(8)  Start Processing this line.
  MONIT.ATTN(8)    Check for ATTN key.
  SHOW.PROGRESS(8)   Put another dot (.) to display.
  PARSE(7)    Parse source line into fields.
     (label field: =BREAK
     (opcode field: LC(5)
     (expression field: 'AB'*16+2)
  LABELS(8)    Process label BREAK.
    >SYT(3)      Put BREAK in symbol table.
  LC+(8)    Update location counter by 7 nibbles.
```

Pass Two Flow:

```
DO.LINE(8)  Finish Processing this line.
  MONIT.ATTN(8)    Check for ATTN key.
  SHOW.PROGRESS(8)   Put another dot (.) to display.
  PARSE(7)    Parse source line into fields.
     (label field: =BREAK
     (opcode field: LC(5)
     (expression field: 'AB'*16+2)
    LABELS(8)      Process label field.
      LABEL.OK?(3)       Verify label BREAK is valid.
      SYT>(3)        Verify value of BREAK same as pass 1.
  PROCESS(8)    Process opcode LC(5)
    BRANCHES(5)      Opcode class is 4 - branches.
      EXPR(5)        Send expression 'AB'*16+2 for evaluation.
        EXPRESSION(4)         Evaluate expression 'AB'*16+2.
          LEXSCAN(4)           Get first token (single quote).
          TERM(4)           Evaluate.  Return 267298.
      FILL.OP(5)       Fill expression  41422 into opcodes 3-7.
  SPIT(8)    Output results of processing this line.
    OUT.OBJ(2)      Output object code:  3422414.
    OUT.LIST(2)      Output listing line to listing
       (assume LISTING is an HP-IL device.)
      OUT.LINE(2)       Send built listing line to listing.
        MONAT.ATTN(8)         Check for ATTN key.
        OUTPUT(SYS)         Send output to HP-IL.
        CRLF(SYS)         Build a carriage return/line feed.
        OUTPUT(SYS)         Send newline to HP-IL.
        &CLEAR(2)         Clear listing line variable.
  LC+(8)    Update location counter by 7 nibbles.
```

Comment Conventions	CHAPTER 4

There are four kinds of comments used in the source listings. The first is a description of each word. The second is a description of what the code is doing. This is embedded inline. The third is the name of the Forth word that the label (in the CON(5)) references. Every headerless assembler word has a name associated with it. The fourth is the condition of the stack after the word on that line has been executed. The rightmost element is on the top of the stack.

4.1 Word Description

Each word in the assembler has a description at its declaration. This description includes a name, what the word does to the data stack, and a brief description of what it does. The name given is the name it would have if it had a header. This is also the name that the third kind of comment references.

4.2 Algorithm description

Throughout the inline code are short, descriptive comments which divide the code into actions. These comments say what the code is doing at this point, or make specific references to other parts of the code when a particularly complicated action occurs.

4.3 Forth Word

Since the assembler is actually coded in SASM assembly language, each reference to another word is made by a label. These labels often aren't descriptive enough, so I've put the Forth word associated with the label in the comment field of the source line. All assembler words are headerless to save space, but each of these has a name (see word description above).

4.4 Stack Description

Following the Forth word is a description of what is left on the stack after the word executes. It usually has the form:

```
( * STR NUM F * )
```

This means there is a string, a number, and a flag on the stack after the word executes. Although there are other items left on the stack besides strings, numbers and flags, a suitable mnemonic was chosen. In this example the flag is on the top of the stack.

Variables — CHAPTER 5

The variables used by the assembler fall into two categories: those kept in the user variable space, and those used strictly during assembly, refered to as temporary variables.

5.1 User Variables

The assembler uses three variables which reside in the FORTHRAM file, and are described in the owner's manual. The first, LISTING(SYS), is a string variable, twenty (20) characters long, that holds the name of the listing file or device. The user sets this variable before an assembly.

The second, PAGESIZE(SYS), is a numeric variable which holds the size of pages for the listing. When a listing device/file is specified, every PAGESIZE(SYS) lines will be a TOF (control L).

The third, VARID(SYS), is used to hold the ID of the general purpose buffer that holds the rest of the variables. The assembler deletes any existing version of a LEX file that it is creating, which causes a configure poll to be issued by the system upon execution of the PURGE command. Therefore, the FORTH system handles the configure poll and saves the buffer whose ID is held in VARID(SYS). Otherwise, all temporary variables would be lost.

5.2 Temporary Variables

When the assembler starts up, VAR.INIT(1) creates a general purpose buffer to hold all the temporary variables. The buffer is created with size 1332 nibbles. These variables act like normal FORTH variables in all respects; except that each reference to a variable finds the general purpose buffer and adds an offset to find the address of the variable.

Numeric Variable Layout Part One

Offset	Variable	Description
000	PASS	Either 1 or 2. Which pass are we in.
005	LC	The location counter.
010	LAST.REQ	A flag to indicate whether the last instruction requires a GOYES/RTNYES.
015	OCCUR	A series of flags: 1 - The listing file is full. 2 - The file type is declared. (LEX, BIN, FORT?? 4 - The symbol table is full. 8 - An error occurred on this source line. 16 - A TOKEN pseudo-op was encountered.
020	KNOWN	Used by SYT>. See Symbol Table chapter.
025	DONE	A flag to indicate END pseudo-op reached.
030	FILETYPE	A variable used to indicate filetype: 1=FORTH, 2=LEX, 3=BIN.
035	SLINE	Current source line number.
040	LNOFF	Offset to end of listing file in RAM.
045	LLINE	Current listing line number.
050	LISTADDR	HP-IL address of listing device.
055	LISTFILE	Address of listing file in RAM.
060	PAGENO	Current page number of listing.
065	TOLIST	Toggle of LIST ON/OFF.
070	OPTYPE	Opcode class of current opcode.
075	OPFLAGS	Opcode flags of current opcode.
080	OPLEN	Length (in nibs) of current opcode.
085	OP	Start of an eighteen variable block, each holdi?? a nibble of object code.
175	FILL.MARK	First nibble that expressions are filled to.
	REG.GROUP	Which register group the opcode is in.
	ARITH.GROUP	Which arithmetic group the opcode is in.
180	FILL.LENGTH	Number of nibbles that expressions are filled t??
185	OPCPOS	Which node in the opcode group we just looked a??
	RTNSTK	Return stack at start of expression evaluation.
190	OPCPTR	Pointer to current opcode entry.
	DATSTK	Data stack at start of expression evaluation.

Numberic Variable Layout Part Two

Offset	Variable	Description
195	OPCTOP	Pointer to beginning of opcode group.
200	VALUE	Value of constant token during expression parsi??
205	TOKEN	Current token during expression parsing.
210	REVERS	Length of expression in LCxxx instructions
215	EXPRCOL	Column of source line that expression field sta??
220	HITOK	Highest lex file TOKEN specified.
225	LOTOK	Lowest lex file TOKEN specified.
230	TENT	Number of ENTRY declarations.
235	KENT	Number of KEY declarations.
240	SYTAD	Start of symbol table (ASMBSY + 37 nibs).
245	SYTSIZE	Number of entries in symbol table.
250	SYTLAST	Last symbol table entry looked at.
255	BTOP	Used in binary search of symbol table.
260	BLOW	Used in binary search of symbol table.
265	BFOUND	Flag indicating symbol found in table.
270	OBJFILE	Start address of object file (for LEX & BIN fil??
275	FILESIZE	Calculated size of object file (after pass 1).
280	LINECNT	Number of dots currently on display.
285	ERRORS	Count of errors this assembly.

String Variable Layout

Offset	Variable	Max size	Description
290	TIT	40	Title line.
374	STIT	40	Subtitle line.
458	SFILE	20	Source file name.
502	SLINE	96	Current source line.
698	LLINE	96	Current listing line.
894	OFILE	20	Object file name.
938	ERR$	50	Error message & temporary storage.
1042	LABEL.FIELD	10	Label field from source line.
1066	OPCODE.FIELD	10	Opcode field from source line.
1090	EXPRESSION.FI	50	Expression field from source line.
1194	IDENTIFIER	10	Value of identifier token during expre?? evaluation.
1218	OLDEXPR	50	Used to save expression.field during m?? expanding pseudo-ops.
1322	-end of general purpose buffer-		

Each variable in the general purpose buffer has a corresponding word in the source file MR&AS1: VARS. Calls to these variables are simply calls to these words.

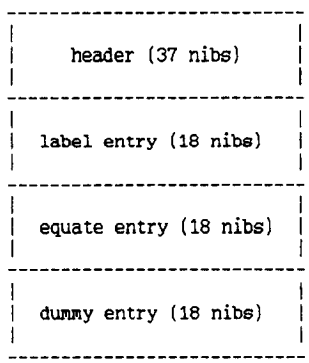
Symbol Table	CHAPTER 6

The symbol table is used to hold all labels and equates. The data is held in a file in the file chain called ASMBSY. All symbol table words are in the source file MR&AS3: Symbol Table.

6.1 Creation of Symbol Table

The Forth word to create the symbol table is SYT.INIT(3). The file SYT.INIT(3) creates is called ASMBSY, a standard TEXT file in the RAM file chain. Originally it has length 9 bytes; each new entry uses another 9 bytes. When first created it has a dummy entry. This entry is the highest possible symbol possible, thus this entry will always be the last entry. A typical symbol table during execution would look like this:

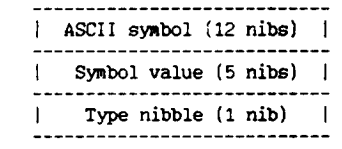
Typical Symbol Table



6.2 Format of Symbol Table Entries

Each entry must contain three items: The symbol itself (with a maximum of 6 characters), its value (a 5 nibble quantity), and a type nibble which indicates a label or an equate. Each entry in the symbol table will look like:

Symbol Table Entry



All symbols in the symbol table are kept in ASCII order. There is no case folding. The dummy entry has ASCII symbol FFFFFFFFFFFF forcing it to the end of the file. It's value & type nibble are FFFFF and F respecively.

The ASCII symbols are always six characters long, right padded with spaces. The symbol value is always an absolute address, even though the listing reports a relative address. This can be achieved because the object file never moves during an assembly. The type nibble can have one of 4 possible meanings: 0 - Equate during pass one, 1 - Label, 2 - Equate during pass two, F - Dummy entry (End of file).

6.3 Important Symbol Table Words

The four key words used for symbol table management are:

SYT.INIT (--) - initializes the symbol table,
>SYT (STR NUM --) - adds symbol STR to symbol table with value NUM,
SYT> (STR -- NUM) - returns the value of symbol STR,
OUT.SYT (--) - outputs the symbol table contents to the listing file??

Note: when SYT> is executed, the variable KNOWN gets set. It can have one of the following values: 0 - symbol not found in symbol table, 1 - symbol found is an equate, -1 - symbol found is a label.

Expression Evaluation — CHAPTER 7

The assembler uses a recursive descent algorithm for evaluating expressions. These words, and all other expression words, are in the source file MR&AS4: EXPRESSIONS.

7.1 Backus-Naur Form

Note: the following symbols are meta-symbols belonging to the BNF formalism, and not symbols of the expression analysis:

```
::=     |     {     }
```

The ::= symbol is read 'is defined as'. The | symbol is read 'or'. The curly brackets denote possible repetition of the enclosed symbols zero or more times. Thus A ::= { B } is a short form for the purely recursive rule A ::= <empty> | AB.

```
<expression> ::= <term>

<term> ::= <factor> { +|- <factor> }

<factor> ::= <boolean> { *|/ <boolean> }

<boolean> ::= <unary-> { &|! <unary-> }

<unary-> ::= -<base> | <base>

<base> ::= DECIMALCONSTANT | HEXADECIMALCONSTANT | 'ASCII' |
           SYMBOL | ( <expression> ) | *
```

7.2 Stack Overflow During Evaluation

One problem generated by this approach occurs when multiple levels of parenthesis are in an expression. The recursion from <base> to <expression> uses seven (7) levels of the return stack plus another seven (7) to do a lexscan. If more than three levels of parenthesis are used, the return stack will overwrite the Terminal Input Buffer.

A routine called DEPTH.CHECK(4) is called at the beginning of each module in the recursive descent chain. It looks to see if the return stack is too close to the TIB(SYS). If it is, it will reset the data and return stacks to their entry-level values and return an error (error 30 'illegal expression'), and a zero on the data stack.

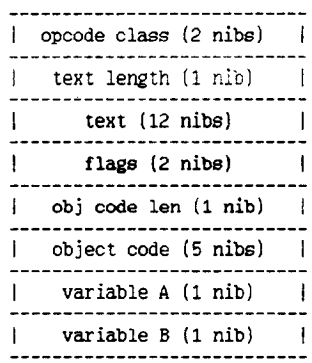
Opcode Lookup & New Opcodes — CHAPTER 8

This chapter describes the opcode tables found in the source file MR&AS0: Opcode Tables, the algorithm used to find opcodes, and how to add new opcodes through hooks in the FORTHRAM file.

8.1 Opcode Tables

The opcode tables are divided into 43 groups (0 to 42), each group having at most 15 opcode entries. Each entry contains information describing the opcode, it's base object code, and some extra information peculiar to that opcode. The layout of an opcode entry is:

Opcode Table Entry



8.2 Description of an Opcode Table Entry

Opcodes are divided into classes based on the processing of the expression field that they require. Internally there are 40 opcode

classes (0 to 39). A table of these classes can be found at the label =aTABLE(8). Each class has a routine which does the required processing.

The text length is a number from 3 through 6, which is simply the number of characters in the text. The text field is right padded with spaces to the 6 character length. This preserves fixed length records.

The flags field has 8 flags which distinguish between certain opcodes in the same class, provides information for the main loop, or tells the lookup routine that this entry is the last in a chain. Here is a description of each flag:

Opcode Flags In Opcode Table Entries

Flags	Meaning if Set
1	The object code length field contains the correct length regardless of any expression field processing. This means that no pass one processing is required.
2	This opcode requires a RTNYES or GOYES instruction following it, otherwise, the main loop will issue a 'requires GOYES/RTNYES' error.
4	This opcode entry is a leaf of the binary tree of opcode entries in this particular group. See the Structure of Opcode Groups.
8	This opcode is a RTNYES or GOYES instruction. The main loop recognizes these instructions and insures a previous test instruction is present.
16	This opcode has an expression imbedded in the object code which is an absolute reference (like GOVLNG, not like GOTO).
32	The expression field of this opcode CAN have quoted spaces. During parsing a space doesn't necessarily mark the end of the expression field.
64	This opcode is one of the gosub opcodes. The opcode has an expression imbedded in the object code which is a relative offset to the address. This offset is calculated from the beginning of the offset with goto opcodes and the end of the offset with gosub opcodes.
128	This opcode has an expression imbedded in the object code; however, if the expression requires more nibbles than the object code can handle, fill as much as will fit.

The object code length and object code are the actual nibbles that get sent to the object file. Of course these are the original nibbles: any processing can modify these nibbles.

Because of the structure of the SASM opcode CON(5), the nibbles get put into memory reversed. Extracting the individual nibbles from this field requires a knowledge of how long the object code is. The

first nibble in memory is the last nibble of the object code. This extraction is done by the word SET.OP.VARS(8) following the comment GET EACH OPCODE SEPARATELY.

The variables are used in three of the opcode classes: register tests, register arithmetic, and branches. In the first two cases, variable A contains the register group to which the opcode belongs. Processing differs for different register groups. In the third case, variable A contains a fill.mark. This is the nibble at which the first nibble of the expression embedded in the object code should start. Variable B contains the fill.length. This is the number of nibbles reserved in the object code for the expression.

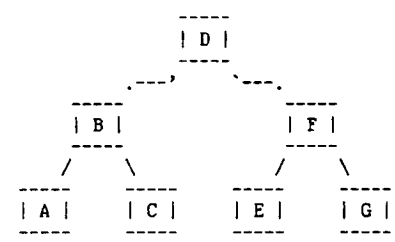
8.3 Hashing function

The hashing function used to get to the opcode's group is done by the word HASH(7). The process is to take the low order three bits from each letter, place them end to end (thus getting an eighteen nibble value) then take this value mod 43. This returns the group.

8.4 Structure of Opcode Groups

The opcodes within a group are arranged in a very specific order. If one were to arrange the opcodes in a binary tree, such that a pre-fix traversal would yield an alphabetic order list, the order becomes apparent. The root node is the first entry. Its two children nodes are the second and third entries. And so forth. Here is a pictoral representation:

Sample Opcode Group



The order within the group would be D-B-F-A-C-E-G. The nodes A, C, E, and G will have the flag 4 set (see above flag descriptions).

8.5 New Opcodes

There is a hook in the assembler for adding new opcodes, or replacing the processing for old opcodes. Whenever the assembler is about to look up an opcode in the tables, it looks at a user variable oPARRT. If this five nibble location is non-zero, it is presumed to contain the CFA of a user-supplied opcode routine. The opcode is placed on the data stack (as a string) and the code identified in oPARRT is EXECUTE(SYS)ed. During processing, when the opcode table entry is scanned, assembler handles all internal opcode classes (0-39); however, if the opcode class is greater than thirty nine (> 39) it puts the opcode clas on the data stack (as a number) and the code identified in oPRORT (in an identical manner to oPARRT) is EXECUTE(SYS)ed. If you're going to do this, be careful: the assembler makes no check that oPRORT is non-zero (if it is zero, a memory reset occurs!) Here are the stack parameters for each routine:

Stack Parameters For Opcode Hook Routines

	1st pass	2nd pass
oPARRT rout.	STR -- FF STR -- PTR TF	STR -- FF STR -- PTR TF
oPRORT rout.	NUM -- LEN	NUM --

8.6 oPARRT Routine (STR -- [PTR] F)

Put the tick (') address of the parsing routine in location 2FC79. Take the string, which is the opcode the assembler is parsing, and check if it is the opcode you're trying to handle. If it isn't, return a false flag on the stack (0). If it is, then you must return a pointer to an opcode table entry (described under Opcode Tables) and a true flag (-1). If you want to handle the processing of the statement (i.e. it isn't a carbon copy of an existing opcode) you will want to make the opcode group greater than thirty nine (>=40). This will cause the second routine, whose tick (') address is in oPRORT, to be evoked during the process stage.

8.7 oPRORT Routine (NUM -- [LEN])

Put the tick (') address of the processing routine in location 2FC7E. The number passed on the stack is the opcode group. It is your job to call the appropriate routines to modify the current opcodes, or to generate the right macro expansions. To call headerless routines from the assembler, an uptodate version of the assembler listings is needed.

The number you return on the stack should only be returned if your routine is called during the first pass. If you don't set flag 1 in the opcode entry, then this routine will get called during pass one. It is your responsibility to return the length (in nibs) of opcodes you will generate during the second pass. If these don't jive, the file chain, labels, and other parts of the system can develop problems. This is not recommended.

8.8 Common Routine Table

When putting in new opcodes, or replacing the processing for old opcodes, you will find many routines are used over and over again. If a temporary scheme is needed, as opposed to a new release, processing routines will need to call internal, headerless words. A table of entry points is included at the end of the ROM for just this purpose. Different releases of the Assembler will keep this table up to date:

Address	Routine whose address is stored there	
EFF00	?PASS=1(5)	(-- F)
EFF05	WRAP.UP.OLD(6)	(--)
EFF0A	SIM.LINE(6)	(STR --)
EFF0F	START.UP.NEW(6)	(STR --)
EFF14	>SYT(3)	(STR NUM --)
EFF19	SYT>(3)	(STR -- NUM)
EFF1E	ERROR(2)	(NUM --)
EFF23	>PAD(2)	(STR -- STR)
EFF28	EXPRESSION	(STR -- NUM)

Address	Variable whose address is stored there	
EFF80	LABEL.FIELD	(-- STR)
EFF85	OPCODE.FIELD	(-- STR)
EFF8A	EXPRESSION.FIELD	(-- STR)

```
EFF8F     FILETYPE          ( -- NUM )
EFF94     KNOWN             ( -- NUM )
EFF99     OLD.EXPRESSION    ( -- STR )
```

I'm not going to go into detail about what these routines do. If you follow the flow of a macro expansion pseudo-op, you'll see most of these in use.

A typical way to compile a call to one of these routines would be as follows:

```
HEX : NULLMACROPSEUDOOP
      [ EFF05  , ]   ( WRAP.UP.OLD )
      [ EFF0F  , ]   ( START.UP.NEW )

    ;
```


Macro Expansion Psuedo-ops	CHAPTER 9

Macro expansion pseudo-ops are a subclass of the pseudo-ops the assembler recognizes. Specifically, they interface with the system in some way. Each macro expansion pseudo-op has just one parameter in the expression field. When interfacing with the system requires more than one parameter, more than one macro expansion pseudo-op will work together.

Each macro expansion pseudo-op generates no code, per se; instead it creates one or more source lines that themselves generate code. Usually the lines generated correspond to specific fields required for interfacing with the system. Sometimes specific lines refer to variables manipulated by other pseudo-ops, e.g. in LEX files, the ID pseudo-op accesses the highest and lowest token declared by the TOKEN pseudo-ops.

The method that macro expansion pseudo-ops use to achieve this effect is somewhat recursive in nature. The processing of a line, as described in the previous Program Flow chapter, is done by the DO.LINE(8) word. This word breakes down into four parts: PARSE(7), PROCESS(8), SPIT(8), and LC+(8). The generation of new source lines occurs during the PROCESS word. At this time the macro expansion pseudo-ops create a new source line, store it in the variable SLINE(1), and call DO.LINE(8). This process is consolidated in SIM.LINE(6).

There is a hitch in this process. If the macro expansion pseudo-op immediately calls SIM.LINE(6), then the processing on the current line is lost. As we said earlier, each macro expansion pseudo-op generates no code. This is because it first sets the variable OPLEN(1) to zero (0) and calls SPIT(8) and LC+(8). This process is consolidated in WRAPUPOLD(6). Similiarly, if the macro expansion pseudo-op called SIM.LINE(6) on the last line, the system would continue with the original DO.LINE(8) and do a SPIT(8) and a LC+(8) again, as the recursion finished. Instead it only does the first two steps: PARSE(7) and PROCESS(8). This process is consolidated in STARTUPNEW(6).

The only other aspect of macro expansion pseudo-ops that isn't covered in the above paragraphs is the handling of labels. The macro expansion pseudo-ops call the words >SYT(3), SYT>(3), and LABEL.OK?(3) directly. This processing doesn't fit in to the nice DO.LINE(8) breakdown, so it is the responsibility of the macro expansion pseudo-ops to handle labels independently.

Assembler Aborting & Ending	CHAPTER 10

Since the assembler has a number of data structures, the aborting and ending routines are alike; however, when the assembler aborts, it must also purge the object and listing files. There are four aborting and ending routines: CLOSEDOWN, CLEANUP, ASSEMBLER ABORT, SHUTDOWN. In this diagram, each routine is shown with the other aborting and ending routines it calls.

Assembler Exit Routines

```
SHUTDOWN(2)
  ASSEM.ABORT(2)
    CLEANUP(2)
      CLOSEDOWN(2)

ASSEM.ABORT(2)
  CLEANUP(2)
    CLOSEDOWN(2)

CLEANUP(2)
  CLOSEDOWN(2)

CLOSEDOWN(2)
```

10.1 SHUTDOWN (NUM --)

This routine requires an error number on the stack. This message is sent to the display and the assembler aborts. All data structures and files associated with the assembly (except source) are removed from the file chain.

This routine is called when the assembler is past the initialization stage and encounters a fatal error.

10.2 ASSEM.ABORT (NUM STR --)

This routine requires an error number and the name of the object file on the stack. Just like SHUTDOWN(2), the error message is sent

to the display and the assembler aborts. All data structures and files associated with the assembly (except source) are removed from the file chain.

This routine is called when the assembler encouters a fatal error, but cannot retrieve the object file name from the temporary variable buffer because the variable buffer is gone.

10.3 CLEANUP (--)

This routine removes the internal files ASMBSY and ASMBOB from the file chain. It is not an abort routine. It closes the source file, kills the variable buffer, resets the ATTN key, and resets the ONERR(SYS) variable.

10.4 CLOSEDOWN (--)

This routine does the latter part of what CLEANUP(2) does. It closes the source file, resets the SCRFIB(SYS) variable, kills the variable buffer, resets the ATTN key, and resets the on error trap.

Out of Memory Condition — CHAPTER 11

When an out of memory condition is recognized, the assembler has to abort (except if listing file fills) and clean up the object file, the symbol table, and the variable buffer. This condition can occur in many ways.

11.1 No Room During Initialization

As the assembler fires up, it must create the symbol table file ASMBSY, the object file ASMBOB, and the variable buffer with id found in the FORTH variable VARID(SYS). If there isn't enough memory the assembler simply quits. This is probably the simplest case of out of memory.

11.2 Symbol Table Fills

Every time a new label is found, an extra eighteen (18) nibbles is required in the symbol table. If ADJUSTF(SYS) cannot add these nibbles the assembler quits with the error 'symbol table full'.

11.3 At the Start of Pass Two

When the assembler begins pass two, it has figured out the number of nibbles required in the object file. If the assembly is FORTH, then it attempts to ALLOT(SYS) this space in the FORTHRAM file. If the assembly is LEX or BIN, it tries to ADJUSTF(SYS) the ASMBOB file by that many nibbles (minus thirty seven (-37) for the file header which already exists). If this fails the assembler quits with the error 'not enough memory for assembler'. This happens in the word OBJ.GROW(2).

11.4 Listing File Fills

This happens only when the FORTH variable LISTING(SYS) is a TEXT file in RAM. As the assembler adds lines to the file, once again a ADJUSTF(SYS) fails. This occurs in the word PRINT (2). The assembler

doesn't quit, it just issues the error 'listing file full'.

11.5 Call to Basic

Whenever a BASICX(SYS), BASIC$(SYS), BASICF(SYS), or BASICI(SYS) is issued, a small buffer is created that builds the basic line to execute. Sometimes this small buffer is enough to cause an out of memory condition. Whenever the assembler makes such a call it sets the FORTH variable ONERR(SYS) so the out of memory will trap to the word MEMAB(1).

The routine MEMAB(1) is required to free up space so the normal shutdown routines can work properly. Since the object and symbol table files need to be purged, a process which requires enough space to make a BASICX call, the normal shutdown routines would cause an infinite loop. To solve this, the shutdown routines are broken down into a part that shutsdown with a variable buffer around, and a part that shutsdown without a variable buffer but with the object file name on the stack. MEMAB(1) gets the object file name onto the stack, kills the variable buffer (thus freeing up space for a BASICX(SYS) call), then calls the second part of the shutdown routines.

Known Bugs	CHAPTER 12

12.1 Word Not Unique

Sometimes during the assembly of FORTH primitives, a warning: word not unique message can appear when it shouldn't. This happens because when a WORD is being assembled, a call to FIND is made. It starts at the current location and searches backwards in the dictionary. When a colon definition is being compiled, the smudge bit is set, thus ensuring the unique message isn't given in reference to the current word. This wasn't necessary in the assembler, since the name field hadn't been put into the dictionary yet. However, when assembling a FORTH primitive in exactly the same place as done previously, the name field from the previous entry remains unchanged. FIND sees this and reports the message.

To recreate this error, this simple file should be assembled:

```
FORTH
WORD   'TEST1'
RTNCC
WORD   'TEST2'
RTNCC
END
```

If you assememble this, FORGET TEST1, and re-assemble, then this message will show up on the word TEST2.

To fix this error, there is a very simple fix. In the source file MR&AS6 at line 959 (just after the comment CHECK IF WORD UNIQUE), add the following lines:

```
CON(5) =LIT          #20          ( * STR NUM * )
CON(5) #20
CON(5) =HERE         HERE         ( * STR NUM ADDR * )
CON(5) =C!           C!           ( * STR * )
```

This will store a two nibble name field with the smudge bit set. This way, FIND will skip over this entry. When processing continues this two nibble field will be overwritten by the correct name field.

12.2 Symbol Table Listing Page

When the symbol table has more entries than will fit on one page, the second page reports invalid values for the entries. What happens is during a page advance BASE is changed to decimal to report the page number correctly. It never gets changed back, hence the values reported for the symbols are reported in decimal not hex. If these values are for a primitive, all addresses within the file require six significat decimal digits, but only five are reported.

To fix this error, in the source file MR&AS3 at line 653 (just after the comment APPEND VALUE TO LISTING LINE), add the following lines:

```
CON(5) =HEX          HEX          ( * STR PTR STR' * )
```

This will reset BASE to hex every time a symbol table value is reported.

Table of Contents

1 Overview

2 Memory Layout During Assembly
2.1 Memory Requirements 2-1

3 Program Flow

4 Comment Conventions
4.1 Word Description 4-1
4.2 Algorithm description 4-1
4.3 Forth Word 4-1
4.4 Stack Description 4-2

5 Variables
5.1 User Variables 5-1
5.2 Temporary Variables 5-1

6 Symbol Table
6.1 Creation of Symbol Table 6-1
6.2 Format of Symbol Table Entries 6-2
6.3 Important Symbol Table Words 6-2

7 Expression Evaluation
7.1 Backus-Naur Form 7-1
7.2 Stack Overflow During Evaluation 7-1

8 Opcode Lookup & New Opcodes
8.1 Opcode Tables 8-1
8.2 Description of an Opcode Table Entry 8-1
8.3 Hashing function 8-3
8.4 Structure of Opcode Groups 8-3
8.5 New Opcodes 8-4
8.6 oPARRT Routine (STR -- [PTR] F) 8-4
8.7 oPRORT Routine (NUM -- [LEN]) 8-5
8.8 Common Routine Table 8-5

9 Macro Expansion Psuedo-ops

10 Assembler Aborting & Ending
10.1 SHUTDOWN (NUM --) 10-1
10.2 ASSEM.ABORT (NUM STR --) 10-1
10.3 CLEANUP (--) 10-2
10.4 CLOSEDOWN (--) 10-2

11 Out of Memory Condition
11.1 No Room During Initialization 11-1
11.2 Symbol Table Fills 11-1

11.3 At the Start of Pass Two 11-1
11.4 Listing File Fills 11-1
11.5 Call to Basic 11-2

12 Known Bugs
12.1 Word Not Unique 12-1
12.2 Symbol Table Listing Page 12-2

```
                TITLE   AS0: FORTH_ASSEMBLER_OPCODE_TABLES
E7400           ABS     #E7400
          zTHIS EQU     *
          zNEXT EQU     #E9900

          ********************************************************
          **
          ** MR&AS0       <840504.1036>
          **
          ** OPCODE LOOKUP TABLES
          **
          ********************************************************
          F.LSET  EQU     01
          F.RETY  EQU     02
          F.TREE  EQU     04
          F.GOYS  EQU     08
          F.ABSL  EQU     16
          F.EXPR  EQU     32
          F.GSUB  EQU     64
          F.PART  EQU    128

```



```
                STITLE  JUMP TABLE
E7400 7D47 *OPTABL CON(5)  OPCG00
          E
E7405 8B57         CON(5)  OPCG01
          E
E740A F277         CON(5)  OPCG02
          E
E740F 0187         CON(5)  OPCG03
          E
E7414 FB87         CON(5)  OPCG04
          E
E7419 40A7         CON(5)  OPCG05
          E
E741E 71B7         CON(5)  OPCG06
          E
E7423 8FB7         CON(5)  OPCG07
          E
E7428 7AC7         CON(5)  OPCG08
          E
E742D E1E7         CON(5)  OPCG09
          E
E7432 DCE7         CON(5)  OPCG10
          E
E7437 C7F7         CON(5)  OPCG11
          E
E743C B208         CON(5)  OPCG12
          E
E7441 E318         CON(5)  OPCG13
          E
E7446 F128         CON(5)  OPCG14
          E
E744B EC28         CON(5)  OPCG15
          E
E7450 3148         CON(5)  OPCG16
          E
E7455 6258         CON(5)  OPCG17
          E
E745A 5D58         CON(5)  OPCG18
          E
E745F 2568         CON(5)  OPCG19
          E
E7464 1078         CON(5)  OPCG20
          E
E7469 E778         CON(5)  OPCG21
          E
E746E D288         CON(5)  OPCG22
          E
E7473 CD88         CON(5)  OPCG23
          E
E7478 B898         CON(5)  OPCG24
          E
E747D 80A8         CON(5)  OPCG25
          E
E7482 B1B8         CON(5)  OPCG26
          E
```

```
     50 E7487 ACB8          CON(5)  OPCG27
              E
     51 E748C BAC8          CON(5)  OPCG28
              E
     52 E7491 82D8          CON(5)  OPCG29
              E
     53 E7496 7DD8          CON(5)  OPCG30
              E
     54 E749B 68E8          CON(5)  OPCG31
              E
     55 E74A0 99F8          CON(5)  OPCG32
              E
     56 E74A5 A709          CON(5)  OPCG33
              E
     57 E74AA D819          CON(5)  OPCG34
              E
     58 E74AF E629          CON(5)  OPCG35
              E
     59 E74B4 1839          CON(5)  OPCG36
              E
     60 E74B9 0349          CON(5)  OPCG37
              E
     61 E74BE FD49          CON(5)  OPCG38
              E
     62 E74C3 2F59          CON(5)  OPCG39
              E
     63 E74C8 3D69          CON(5)  OPCG40
              E
     64 E74CD 0579          CON(5)  OPCG41
              E
     65 E74D2 1389          CON(5)  OPCG42
              E
```



```
     66                     STITLE  HASH TO GROUP 0
     67 E74D7       OPCG00
     68 E74D7 50            CON(2)  5       FIXED
     69 E74D9 4             CON(1)  4       OPCODE MNEMONIC
     70 E74DA 3435          NIBASC  \CSLC  \
              C434
              0202
     71 E74E6 10            CON(2)  (F.LSET)
     72 E74E8 3             CON(1)  3       OPCODE
     73 E74E9 2180          CON(5)  #812
              0
     74 E74EE 00            CON(2)  0       NO VARIABLES
     75               *
     76               *
     77 E74F0 30            CON(2)  3       REGISTER LOGIC
     78 E74F2 5             CON(1)  5       OPCODE MNEMONIC
     79 E74F3 34D3          NIBASC  \C=C!B \
              3412
              2402
     80 E74FF 10            CON(2)  (F.LSET)
     81 E7501 4             CON(1)  4       OPCODE
     82 E7502 D0E0          CON(5)  #0E0D
              0
     83 E7507 00            CON(2)  0       NO VARIABLES
     84               *
     85               *
     86 E7509 A0            CON(2)  10      SET POINTER
     87 E750B 3             CON(1)  3       OPCODE MNEMONIC
     88 E750C 05D3          NIBASC  \P=C   \
              3402
              0202
     89 E7518 12            CON(2)  (F.LSET)!(F.EXPR)
     90 E751A 4             CON(1)  4       Q CODE
     91 E751B 0D08          CON(5)  #80D0
              0
     92 E7520 00            CON(2)  0       NO VARIABLES
     93               *
     94               *
     95 E7522 10            CON(2)  1       REGISTER TESTS
     96 E7524 4             CON(1)  4       OPCODE MNEMONIC
     97 E7525 F314          NIBASC  \?A>B  \
              E324
              0202
     98 E7531 30            CON(2)  (F.LSET)!(F.RETY)
     99 E7533 3             CON(1)  3       OPCODE
    100 E7534 0080          CON(5)  #800
              0
    101 E7539 2             CON(1)  2       GROUP B
    102 E753A 0             CON(1)  0
    103              *
    104              *
    105 E753B 50            CON(2)  5       FIXED
    106 E753D 5             CON(1)  5       OPCODE MNEMONIC
    107 E753E 3425          NIBASC  \CR1EX \
              1354
```

```
          8502
108 E754A 50          CON(2)  (F.LSET)!(F.TREE)
109 E754C 3           CON(1)  3       OPCODE
110 E754D 9210        CON(5)  #129
          0
111 E7552 00          CON(2)  0       NO VARIABLES
112             *
113             *
114 E7554 50          CON(2)  5       FIXED
115 E7556 5           CON(1)  5       OPCODE MNEMONIC
116 E7557 4413        NIBASC  \D1=CS \
          D334
          3502
117 E7563 50          CON(2)  (F.LSET)!(F.TREE)
118 E7565 3           CON(1)  3       OPCODE
119 E7566 D310        CON(5)  #13D
          0
120 E756B 00          CON(2)  0       NO VARIABLES
121             *
122             *
123 E756D 50          CON(2)  5       FIXED
124 E756F 6           CON(1)  6       OPCODE MNEMONIC
125 E7570 3554        NIBASC  \SETDEC\
          4544
          5434
126 E757C 50          CON(2)  (F.LSET)!(F.TREE)
127 E757E 2           CON(1)  2       OPCODE
128 E757F 5000        CON(5)  #05
          0
129 E7584 00          CON(2)  0       NO VARIABLES
130             *
131             *
132 E7586 00  =NULENT CON(2)  0       NULL
133 E7588 0           CON(1)  0       OPCODE MNEMONIC
134 E7589 0202        NIBASC  \      \
          0202
          0202
135 E7595 50          CON(2)  (F.LSET)!(F.TREE)
136 E7597 0           CON(1)  0       OPCODE
137 E7598 0000        CON(5)  0
          0
138 E759D 00          CON(2)  0       NO VARIABLES
139             *
140             *
141 E759F 50          CON(2)  5       FIXED
142 E75A1 5           CON(1)  5       OPCODE MNEMONIC
143 E75A2 1425        NIBASC  \AR0EX \
          0354
          8502
144 E75AE 50          CON(2)  (F.LSET)!(F.TREE)
145 E75B0 3           CON(1)  3       OPCODE
146 E75B1 0210        CON(5)  #120
          0
147 E75B6 00          CON(2)  0       NO VARIABLES
```



```
148                   STITLE  HASH TO GROUP 1
149 E75B8       OPCG01
150 E75B8 D0          CON(2)  13      DATA TRANSFER
151 E75BA 6           CON(1)  6       OPCODE MNEMONIC
152 E75BB 4414        NIBASC  \DAT0=A\
          4503
          D314
153 E75C7 02          CON(2)  (F.EXPR)
154 E75C9 4           CON(1)  4       OPCODE
155 E75CA 0051        CON(5)  #1500
          0
156 E75CF 00          CON(2)  0       NO VARIABLES
157             *
158             *
159 E75D1 10          CON(2)  1       REGISTER TESTS
160 E75D3 4           CON(1)  4       OPCODE MNEMONIC
161 E75D4 F334        NIBASC  \?C>A  \
          E314
          0202
162 E75E0 30          CON(2)  (F.LSET)!(F.RETY)
163 E75E2 3           CON(1)  3       OPCODE
164 E75E3 2080        CON(5)  #802
          0
165 E75E8 2           CON(1)  2       GROUP B
166 E75E9 0           CON(1)  0
167             *
168             *
169 E75EA 40          CON(2)  4       BRANCHES
170 E75EC 5           CON(1)  5       OPCODE MNEMONIC
171 E75ED 74F4        NIBASC  \GOYES \
          9554
          3502
172 E75F9 92          CON(2)  (F.LSET)!(F.GOYS)!(F.EXPR)
173 E75FB 2           CON(1)  2       OPCODE
174 E75FC 0000        CON(5)  #00
          0
175 E7601 1           CON(1)  1       FILL.MARK
176 E7602 2           CON(1)  2       FILL.LENGTH
177             *
178             *
179 E7603 10          CON(2)  1       REGISTER TESTS
180 E7605 5           CON(1)  5       OPCODE MNEMONIC
181 E7606 F314        NIBASC  \?A>=C \
          E3D3
          3402
182 E7612 30          CON(2)  (F.LSET)!(F.RETY)
183 E7614 3           CON(1)  3       OPCODE
184 E7615 E080        CON(5)  #80E
          0
185 E761A 2           CON(1)  2       GROUP B
186 E761B 0           CON(1)  0
187             *
188             *
189 E761C 12          CON(2)  33      PSEUDO-OP
190 E761E 4           CON(1)  4       OPCODE MNEMONIC
```

```
191 E761F 3484          NIBASC  \CHAR  \
          1425
          0202
192 E762B 12            CON(2)  (F.LSET)!(F.EXPR)
193 E762D 1             CON(1)  1
194 E762E 0000          CON(5)  #0
          0
195 E7633 00            CON(2)  0       NO VARIABLES
196             *
197             *
198 E7635 40            CON(2)  4       BRANCHES
199 E7637 4             CON(1)  4       OPCODE MNEMONIC
200 E7638 74F4          NIBASC  \GONC  \
          E434
          0202
201 E7644 12            CON(2)  (F.LSET)!(F.EXPR)
202 E7646 3             CON(1)  3       OPCODE
203 E7647 0050          CON(5)  #500
          0
204 E764C 2             CON(1)  2       FILL.MARK
205 E764D 2             CON(1)  2       FILL.LENGTH
206             *
207             *
208 E764E 50            CON(2)  5       FIXED
209 E7650 3             CON(1)  3       OPCODE MNEMONIC
210 E7651 2545          NIBASC  \RTI   \
          9402
          0202
211 E765D 10            CON(2)  (F.LSET)
212 E765F 2             CON(1)  2       OPCODE
213 E7660 F000          CON(5)  #0F
          0
214 E7665 00            CON(2)  0       NO VARIABLES
215             *
216             *
217 E7667 10            CON(2)  1       REGISTER TESTS
218 E7669 5             CON(1)  5       OPCODE MNEMONIC
219 E766A F314          NIBASC  \?A<=B \
          C3D3
          2402
220 E7676 70            CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
221 E7678 3             CON(1)  3       OPCODE
222 E7679 C080          CON(5)  #80C
          0
223 E767E 2             CON(1)  2       GROUP B
224 E767F 0             CON(1)  0
225             *
226             *
227 E7680 10            CON(2)  1       REGISTER TESTS
228 E7682 4             CON(1)  4       OPCODE MNEMONIC
229 E7683 F324          NIBASC  \?B#C  \
          3234
          0202
230 E768F 70            CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
231 E7691 3             CON(1)  3       OPCODE
```



```
232 E7692 5080          CON(5)  #805
          0
233 E7697 1             CON(1)  1       GROUP A
234 E7698 0             CON(1)  0
235             *
236             *
237 E7699 20            CON(2)  2       REGISTER ARITHMATIC
238 E769B 5             CON(1)  5       OPCODE MNEMONIC
239 E769C 24D3          NIBASC  \B=B+B \
          24B2
          2402
240 E76A8 50            CON(2)  (F.LSET)!(F.TREE)
241 E76AA 3             CON(1)  3       OPCODE
242 E76AB 5000          CON(5)  #005
          0
243 E76B0 1             CON(1)  1       GROUP C
244 E76B1 0             CON(1)  0
245             *
246             *
247 E76B2 50            CON(2)  5       FIXED
248 E76B4 5             CON(1)  5       OPCODE MNEMONIC
249 E76B5 4413          NIBASC  \D1=AS \
          D314
          3502
250 E76C1 50            CON(2)  (F.LSET)!(F.TREE)
251 E76C3 3             CON(1)  3       OPCODE
252 E76C4 8310          CON(5)  #138
          0
253 E76C9 00            CON(2)  0       NO VARIABLES
254             *
255             *
256 E76CB 61            CON(2)  22      EQU PSUEDO-OP
257 E76CD 3             CON(1)  3       OPCODE MNEMONIC
258 E76CE 5415          NIBASC  \EQU   \
          5502
          0202
259 E76DA 42            CON(2)  (F.TREE)!(F.EXPR)
260 E76DC 0             CON(1)  0       OPCODE
261 E76DD 0000          CON(5)  #0
          0
262 E76E2 00            CON(2)  0       NO VARIABLES
263             *
264             *
265 E76E4 40            CON(2)  4       BRANCHES
266 E76E6 6             CON(1)  6       OPCODE MNEMONIC
267 E76E7 74F4          NIBASC  \GOSBVL\
          3524
          65C4
268 E76F3 53            CON(2)  (F.LSET)!(F.TREE)!(F.ABSL)!(F.EXPR)
269 E76F5 7             CON(1)  7       OPCODE
270 E76F6 000F          CON(5)  #8F000
          8
271 E76FB 3             CON(1)  3       FILL.MARK
272 E76FC 5             CON(1)  5       FILL.LENGTH
273             *
```

```
274                *
275 E76FD 50               CON(2)  5       FIXED
276 E76FF 4                CON(1)  4       OPCODE MNEMONIC
277 E7700 E4F4             NIBASC  \NOP4  \
          0543
          0202
278 E770C 50               CON(2)  (F.LSET)!(F.TREE)
279 E770E 4                CON(1)  4       OPCODE
280 E770F 0036             CON(5)  #6300
          0
281 E7714 00               CON(2)  0       NO VARIABLES
282                *
283                *
284 E7716 50               CON(2)  5       FIXED
285 E7718 4                CON(1)  4       OPCODE MNEMONIC
286 E7719 2545             NIBASC  \RTNC  \
          E434
          0202
287 E7725 50               CON(2)  (F.LSET)!(F.TREE)
288 E7727 3                CON(1)  3       OPCODE
289 E7728 0040             CON(5)  #400
          0
290 E772D 00               CON(2)  0       NO VARIABLES
```



```
291                *       STITLE  HASH TO GROUP 2
292 E772F       OPCG02
293 E772F 20               CON(2)  2       REGISTER ARITHMETIC
294 E7731 3                CON(1)  3       OPCODE MNEMONIC
295 E7732 34D3             NIBASC  \C=D   \
          4402
          0202
296 E773E 00               CON(2)  0
297 E7740 3                CON(1)  3       OPCODE
298 E7741 B000             CON(5)  #00B
          0
299 E7746 2                CON(1)  2       GROUP D
300 E7747 0                CON(1)  0
301                *
302                *
303 E7748 20               CON(2)  2       REGISTER ARITHMETIC
304 E774A 5                CON(1)  5       OPCODE MNEMONIC
305 E774B 14D3             NIBASC  \A=A-B \
          14D2
          2402
306 E7757 00               CON(2)  0
307 E7759 3                CON(1)  3       OPCODE
308 E775A 0000             CON(5)  #000
          0
309 E775F 3                CON(1)  3       GROUP E
310 E7760 0                CON(1)  0
311                *
312                *
313 E7761 50               CON(2)  5       FIXED
314 E7763 5                CON(1)  5       OPCODE MNEMONIC
315 E7764 2545             NIBASC  \RTNNC \
          E4E4
          3402
316 E7770 10               CON(2)  (F.LSET)
317 E7772 3                CON(1)  3       OPCODE
318 E7773 0050             CON(5)  #500
          0
319 E7778 00               CON(2)  0       NO VARIABLES
320                *
321                *
322 E777A 60               CON(2)  6       HARDWARE TESTS
323 E777C 5                CON(1)  5       OPCODE MNEMONIC
324 E777D F3D4             NIBASC  \?MP=0 \
          05D3
          0302
325 E7789 30               CON(2)  (F.LSET)!(F.RETY)
326 E778B 3                CON(1)  3       OPCODE
327 E778C 8380             CON(5)  #838
          0
328 E7791 00               CON(2)  0       NO VARIABLES
329                *
330                *
331 E7793 20               CON(2)  2       REGISTER ARITHMETIC
332 E7795 3                CON(1)  3       OPCODE MNEMONIC
333 E7796 14D3             NIBASC  \A=C   \
```

```
            3402
            0202
    334 E77A2 40          CON(2)  (F.TREE)
    335 E77A4 3           CON(1)  3       OPCODE
    336 E77A5 A000        CON(5)  #00A
            0
    337 E77AA 2           CON(1)  2       GROUP D
    338 E77AB 0           CON(1)  0
    339             *
    340             *
    341 E77AC 50          CON(2)  5       FIXED
    342 E77AE 4           CON(1)  4       OPCODE MNEMONIC
    343 E77AF 4435        NIBASC  \DSLC  \
            C434
            0202
    344 E77BB 50          CON(2)  (F.LSET)!(F.TREE)
    345 E77BD 3           CON(1)  3       OPCODE
    346 E77BE 3180        CON(5)  #813
            0
    347 E77C3 00          CON(2)  0       NO VARIABLES
    348             *
    349             *
    350 E77C5 70          CON(2)  7       RETURN YES
    351 E77C7 6           CON(1)  6       OPCODE MNEMONIC
    352 E77C8 2545        NIBASC  \RTNYES\
            E495
            5435
    353 E77D4 D0          CON(2)  (F.LSET)!(F.GOYS)!(F.TREE)
    354 E77D6 2           CON(1)  2       OPCODE
    355 E77D7 0000        CON(5)  #00
            0
    356 E77DC 00          CON(2)  0       NO VARIABLES
    357             *
    358             *
    359 E77DE 10          CON(2)  1       REGISTER TESTS
    360 E77E0 4           CON(1)  4       OPCODE MNEMONIC
    361 E77E1 F324        NIBASC  \?B#A  \
            3214
            0202
    362 E77ED 70          CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
    363 E77EF 3           CON(1)  3       OPCODE
    364 E77F0 4080        CON(5)  #804
            0
    365 E77F5 1           CON(1)  1       GROUP A
    366 E77F6 0           CON(1)  0
    367             *
    368             *
    369 E77F7 80          CON(2)  8       POINTER TESTS
    370 E77F9 3           CON(1)  3       OPCODE MNEMONIC
    371 E77FA F305        NIBASC  \?P#   \
            3202
            0202
    372 E7806 72          CON(2)  (F.LSET)!(F.RETY)!(F.TREE)!(F.EXPR)
    373 E7808 3           CON(1)  3       OPCODE
    374 E7809 0880        CON(5)  #880
```



```
            0
    375 E780E 00          CON(2)  0       NO VARIABLES
```

```
376                     STITLE  HASH TO GROUP 3
377 E7810       OPCG03
378 E7810 D0            CON(2)  13      DATA TRANSFER
379 E7812 6             CON(1)  6       OPCODE MNEMONIC
380 E7813 4414          NIBASC  \DAT0=C\
          4503
          D334
381 E781F 02            CON(2)  (F.EXPR)
382 E7821 4             CON(1)  4       OPCODE
383 E7822 0451          CON(5)  #1540
          0
384 E7827 00            CON(2)  0       NO VARIABLES
385             *
386             *
387 E7829 20            CON(2)  2       REGISTER ARITHMETIC
388 E782B 5             CON(1)  5       OPCODE MNEMONIC
389 E782C 14D3          NIBASC  \A=A+B \
          14B2
          2402
390 E7838 00            CON(2)  0
391 E783A 3             CON(1)  3       OPCODE
392 E783B 0000          CON(5)  #000
          0
393 E7840 1             CON(1)  1       GROUP C
394 E7841 0             CON(1)  0
395             *
396             *
397 E7842 B1            CON(2)  27      IMMEDIATE FORTH WORD
398 E7844 5             CON(1)  5       OPCODE MNEMONIC
399 E7845 75F4          NIBASC  \WORDI \
          2544
          9402
400 E7851 02            CON(2)  (F.EXPR)
401 E7853 0             CON(1)  0       OPCODE
402 E7854 0000          CON(5)  #0
          0
403 E7859 00            CON(2)  0       NO VARIABLES
404             *
405             *
406 E785B 10            CON(2)  1       REGISTER TESTS
407 E785D 4             CON(1)  4       OPCODE MNEMONIC
408 E785E F344          NIBASC  \?D#0  \
          3203
          0202
409 E786A 70            CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
410 E786C 3             CON(1)  3       OPCODE
411 E786D F080          CON(5)  #80F
          0
412 E7872 1             CON(1)  1       GROUP A
413 E7873 0             CON(1)  0
414             *
415             *
416 E7874 20            CON(2)  2       REGISTER ARITHMETIC
417 E7876 5             CON(1)  5       OPCODE MNEMONIC
418 E7877 34D3          NIBASC  \C=D+C \
```



```
          44B2
          3402
419 E7883 40            CON(2)  (F.TREE)
420 E7885 3             CON(1)  3       OPCODE
421 E7886 B000          CON(5)  #00B
          0
422 E788B 1             CON(1)  1       GROUP C
423 E788C 0             CON(1)  0
424             *
425             *
426 E788D D1            CON(2)  29      PSEUDO-OP
427 E788F 2             CON(1)  2       OPCODE MNEMONIC
428 E7890 9444          NIBASC  \ID    \
          0202
          0202
429 E789C 42            CON(2)  (F.TREE)!(F.EXPR)
430 E789E 0             CON(1)  0       OPCODE
431 E789F 0000          CON(5)  #0
          0
432 E78A4 00            CON(2)  0       NO VARIABLES
433             *
434             *
435 E78A6 00            CON(2)  0       NULL
436 E78A8 0             CON(1)  0       OPCODE MNEMONIC
437 E78A9 0202          NIBASC  \      \
          0202
          0202
438 E78B5 40            CON(2)  (F.TREE)
439 E78B7 0             CON(1)  0       OPCODE
440 E78B8 0000          CON(5)  #0
          0
441 E78BD 00            CON(2)  0       NO VARIABLES
```

```
                        STITLE  HASH TO GROUP 4
E78BF       OPCG04
E78BF 30                CON(2)  3       REGISTER LOGIC
E78C1 5                 CON(1)  5       OPCODE MNEMONIC
E78C2 34D3              NIBASC  \C=D!C \
      4412
      3402
E78CE 10                CON(2)  (F.LSET)
E78D0 4                 CON(1)  4       OPCODE
E78D1 F0E0              CON(5)  #0E0F
      0
E78D6 00                CON(2)  0       NO VARIABLES
            *
            *
E78D8 50                CON(2)  5       FIXED
E78DA 4                 CON(1)  4       OPCODE MNEMONIC
E78DB 14D3              NIBASC  \A=R4  \
      2543
      0202
E78E7 10                CON(2)  (F.LSET)
E78E9 3                 CON(1)  3       OPCODE
E78EA 4110              CON(5)  #114
      0
E78EF 00                CON(2)  0       NO VARIABLES
            *
            *
E78F1 40                CON(2)  4       BRANCHES
E78F3 5                 CON(1)  5       OPCODE MNEMONIC
E78F4 C434              NIBASC  \LC(4) \
      8243
      9202
E7900 1B                CON(2)  (F.LSET)!(F.ABSL)!(F.EXPR)!(F.PART)
E7902 6                 CON(1)  6       OPCODE
E7903 0003              CON(5)  #33000
      3
E7908 3                 CON(1)  3       FILL.MARK
E7909 4                 CON(1)  4       FILL.LENGTH
            *
            *
E790A 10                CON(2)  1       REGISTER TESTS
E790C 5                 CON(1)  5       OPCODE MNEMONIC
E790D F324              NIBASC  \?B<=A \
      C3D3
      1402
E7919 30                CON(2)  (F.LSET)!(F.RETY)
E791B 3                 CON(1)  3       OPCODE
E791C 8080              CON(5)  #808
      0
E7921 2                 CON(1)  2       GROUP B
E7922 0                 CON(1)  0
            *
            *
E7923 50                CON(2)  5       FIXED
E7925 4                 CON(1)  4       OPCODE MNEMONIC
E7926 1435              NIBASC  \ASRC  \
```



```
      2534
      0202
E7932 10                CON(2)  (F.LSET)
E7934 3                 CON(1)  3       OPCODE
E7935 4180              CON(5)  #814
      0
E793A 00                CON(2)  0       NO VARIABLES
            *
            *
E793C 50                CON(2)  5       FIXED
E793E 5                 CON(1)  5       OPCODE MNEMONIC
E793F 3425              NIBASC  \CR0EX \
      0354
      8502
E794B 10                CON(2)  (F.LSET)
E794D 3                 CON(1)  3       OPCODE
E794E 8210              CON(5)  #128
      0
E7953 00                CON(2)  0       NO VARIABLES
            *
            *
E7955 50                CON(2)  5       FIXED
E7957 4                 CON(1)  4       OPCODE MNEMONIC
E7958 2503              NIBASC  \R0=C  \
      D334
      0202
E7964 50                CON(2)  (F.LSET)!(F.TREE)
E7966 3                 CON(1)  3       OPCODE
E7967 8010              CON(5)  #108
      0
E796C 00                CON(2)  0       NO VARIABLES
            *
            *
E796E 10                CON(2)  1       REGISTER TESTS
E7970 4                 CON(1)  4       OPCODE MNEMONIC
E7971 F314              NIBASC  \?A=B  \
      D324
      0202
E797D 70                CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
E797F 3                 CON(1)  3       OPCODE
E7980 0080              CON(5)  #800
      0
E7985 1                 CON(1)  1       GROUP A
E7986 0                 CON(1)  0
            *
            *
E7987 30                CON(2)  3       REGISTER LOGIC
E7989 5                 CON(1)  5       OPCODE MNEMONIC
E798A 14D3              NIBASC  \A=A!B \
      1412
      2402
E7996 50                CON(2)  (F.LSET)!(F.TREE)
E7998 4                 CON(1)  4       OPCODE
E7999 80E0              CON(5)  #0E08
      0
```

```
525 E799E 00          CON(2)  0       NO VARIABLES
526           *
527           *
528 E79A0 20          CON(2)  2       FIXED
529 E79A2 4           CON(1)  4       OPCODE MNEMONIC
530 E79A3 1424        NIBASC  \ABEX  \
          5485
          0202
531 E79AF 40          CON(2)  (F.TREE)
532 E79B1 3           CON(1)  3       OPCODE
533 E79B2 C000        CON(5)  #00C
          0
534 E79B7 2           CON(1)  2       GROUP D
535 E79B8 0           CON(1)  0
536           *
537           *
538 E79B9 20          CON(2)  2       REGISTER ARITHMETIC
539 E79BB 3           CON(1)  3       OPCODE MNEMONIC
540 E79BC 24D3        NIBASC  \B=C   \
          3402
          0202
541 E79C8 40          CON(2)  (F.TREE)
542 E79CA 3           CON(1)  3       OPCODE
543 E79CB 5000        CON(5)  #005
          0
544 E79D0 2           CON(1)  2       GROUP D
545 E79D1 0           CON(1)  0
546           *
547           *
548 E79D2 50          CON(2)  5       FIXED
549 E79D4 4           CON(1)  4       OPCODE MNEMONIC
550 E79D5 34D3        NIBASC  \C=ST  \
          3545
          0202
551 E79E1 50          CON(2)  (F.LSET)!(F.TREE)
552 E79E3 2           CON(1)  2       OPCODE
553 E79E4 9000        CON(5)  #09
          0
554 E79E9 00          CON(2)  0       NO VARIABLES
555           *
556           *
557 E79EB 20          CON(2)  2       REGISTER ARITHMETIC
558 E79ED 5           CON(1)  5       OPCODE MNEMONIC
559 E79EE 44D3        NIBASC  \D=D-C \
          44D2
          3402
560 E79FA 40          CON(2)  (F.TREE)
561 E79FC 3           CON(1)  3       OPCODE
562 E79FD 3000        CON(5)  #003
          0
563 E7A02 3           CON(1)  3       GROUP E
564 E7A03 0           CON(1)  0
```



```
565           STITLE  HASH TO GROUP 5
566 E7A04 OPCG05
567 E7A04 40          CON(2)  4       BRANCHES
568 E7A06 5           CON(1)  5       OPCODE MNEMONIC
569 E7A07 C434        NIBASC  \LC(2) \
          8223
          9202
570 E7A13 1B          CON(2)  (F.LSET)!(F.ABSL)!(F.EXPR)!(F.PART)
571 E7A15 4           CON(1)  4       OPCODE
572 E7A16 0013        CON(5)  #3100
          0
573 E7A1B 3           CON(1)  3       FILL.MARK
574 E7A1C 2           CON(1)  2       FILL.LENGTH
575           *
576           *
577 E7A1D 20          CON(2)  2       REGISTER ARITHMETIC
578 E7A1F 5           CON(1)  5       OPCODE MNEMONIC
579 E7A20 24D3        NIBASC  \B=A+B \
          14B2
          2402
580 E7A2C 00          CON(2)  0
581 E7A2E 3           CON(1)  3       OPCODE
582 E7A2F 8000        CON(5)  #008
          0
583 E7A34 1           CON(1)  1       GROUP C
584 E7A35 0           CON(1)  0
585           *
586           *
587 E7A36 50          CON(2)  5       FIXED
588 E7A38 4           CON(1)  4       OPCODE MNEMONIC
589 E7A39 2503        NIBASC  \R0=A  \
          D314
          0202
590 E7A45 10          CON(2)  (F.LSET)
591 E7A47 3           CON(1)  3       OPCODE
592 E7A48 0010        CON(5)  #100
          0
593 E7A4D 00          CON(2)  0       NO VARIABLES
594           *
595           *
596 E7A4F 10          CON(2)  1       REGISTER TESTS
597 E7A51 4           CON(1)  4       OPCODE MNEMONIC
598 E7A52 F334        NIBASC  \?C=A  \
          D314
          0202
599 E7A5E 30          CON(2)  (F.LSET)!(F.RETY)
600 E7A60 3           CON(1)  3       OPCODE
601 E7A61 2080        CON(5)  #802
          0
602 E7A66 1           CON(1)  1       GROUP A
603 E7A67 0           CON(1)  0
604           *
605           *
606 E7A68 40          CON(2)  4       BRANCHES
607 E7A6A 6           CON(1)  6       OPCODE MNEMONIC
```

```
608 E7A6B 4403          NIBASC  \D0=(2)\
          D382
          2392
609 E7A77 1B            CON(2)  (F.LSET)!(F.ABSL)!(F.EXPR)!(F.PART)
610 E7A79 4             CON(1)  4       OPCODE
611 E7A7A 0091          CON(5)  #1900
          0
612 E7A7F 3             CON(1)  3       FILL.MARK
613 E7A80 2             CON(1)  2       FILL.LENGTH
614             *
615             *
616 E7A81 71            CON(2)  23      LIST PSEUDO-OP
617 E7A83 4             CON(1)  4       OPCODE MNEMONIC
618 E7A84 C494          NIBASC  \LIST  \
          3545
          0202
619 E7A90 50            CON(2)  (F.LSET)!(F.TREE)
620 E7A92 0             CON(1)  0       OPCODE
621 E7A93 0000          CON(5)  #0
          0
622 E7A98 00            CON(2)  0       NO VARIABLES
623             *
624             *
625 E7A9A 50            CON(2)  5       FIXED
626 E7A9C 4             CON(1)  4       OPCODE MNEMONIC
627 E7A9D 2543          NIBASC  \R4=C  \
          D334
          0202
628 E7AA9 50            CON(2)  (F.LSET)!(F.TREE)
629 E7AAB 3             CON(1)  3       OPCODE
630 E7AAC C010          CON(5)  #10C
          0
631 E7AB1 00            CON(2)  0       NO VARIABLES
632             *
633             *
634 E7AB3 10            CON(2)  1       REGISTER TESTS
635 E7AB5 4             CON(1)  4       OPCODE MNEMONIC
636 E7AB6 F314          NIBASC  \?A=0  \
          D303
          0202
637 E7AC2 70            CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
638 E7AC4 3             CON(1)  3       OPCODE
639 E7AC5 8080          CON(5)  #808
          0
640 E7ACA 1             CON(1)  1       GROUP A
641 E7ACB 0             CON(1)  0
642             *
643             *
644 E7ACC 50            CON(2)  5       FIXED
645 E7ACE 4             CON(1)  4       OPCODE MNEMONIC
646 E7ACF 14D3          NIBASC  \A=R2  \
          2523
          0202
647 E7ADB 50            CON(2)  (F.LSET)!(F.TREE)
648 E7ADD 3             CON(1)  3       OPCODE
```



```
649 E7ADE 2110          CON(5)  #112
          0
650 E7AE3 00            CON(2)  0       NO VARIABLES
651             *
652             *
653 E7AE5 50            CON(2)  5       FIXED
654 E7AE7 5             CON(1)  5       OPCODE MNEMONIC
655 E7AE8 2455          NIBASC  \BUSCC \
          3534
          3402
656 E7AF4 50            CON(2)  (F.LSET)!(F.TREE)
657 E7AF6 3             CON(1)  3       OPCODE
658 E7AF7 B080          CON(5)  #80B
          0
659 E7AFC 00            CON(2)  0       NO VARIABLES
660             *
661             *
662 E7AFE 20            CON(2)  2       REGISTER ARITHMETIC
663 E7B00 5             CON(1)  5       OPCODE MNEMONIC
664 E7B01 44D3          NIBASC  \D=D+C \
          44B2
          3402
665 E7B0D 40            CON(2)  (F.TREE)
666 E7B0F 3             CON(1)  3       OPCODE
667 E7B10 3000          CON(5)  #003
          0
668 E7B15 1             CON(1)  1       GROUP C
669 E7B16 0             CON(1)  0
```

```
                        STITLE  HASH TO GROUP 6
E7B17       OPCG06
E7B17 40                CON(2)  4       BRANCHES
E7B19 6                 CON(1)  6       OPCODE MNEMONIC
E7B1A 34F4              NIBASC  \CON(4)\
      E482
      4392
E7B26 13                CON(2)  (F.LSET)!(F.ABSL)!(F.EXPR)
E7B28 4                 CON(1)  4       OPCODE
E7B29 0000              CON(5)  #0
      0
E7B2E 1                 CON(1)  1       FILL.MARK
E7B2F 4                 CON(1)  4       FILL.LENGTH
                *
                *
E7B30 30                CON(2)  3       REGISTER LOGIC
E7B32 5                 CON(1)  5       OPCODE MNEMONIC
E7B33 24D3              NIBASC  \B=A!B \
      1412
      2402
E7B3F 10                CON(2)  (F.LSET)
E7B41 4                 CON(1)  4       OPCODE
E7B42 C0E0              CON(5)  #0E0C
      0
E7B47 00                CON(2)  0       NO VARIABLES
                *
                *
E7B49 50                CON(2)  5       FIXED
E7B4B 4                 CON(1)  4       OPCODE MNEMONIC
E7B4C 2543              NIBASC  \R4=A  \
      D314
      0202
E7B58 10                CON(2)  (F.LSET)
E7B5A 3                 CON(1)  3       OPCODE
E7B5B 4010              CON(5)  #104
      0
E7B60 00                CON(2)  0       NO VARIABLES
                *
                *
E7B62 50                CON(2)  5       FIXED
E7B64 4                 CON(1)  4       OPCODE MNEMONIC
E7B65 14D3              NIBASC  \A=R0  \
      2503
      0202
E7B71 10                CON(2)  (F.LSET)
E7B73 3                 CON(1)  3       OPCODE
E7B74 0110              CON(5)  #110
      0
E7B79 00                CON(2)  0       NO VARIABLES
                *
                *
E7B7B 50                CON(2)  5       FIXED
E7B7D 4                 CON(1)  4       OPCODE MNEMONIC
E7B7E 2435              NIBASC  \BSRC  \
      2534
```



```
      0202
E7B8A 50                CON(2)  (F.LSET)!(F.TREE)
E7B8C 3                 CON(1)  3       OPCODE
E7B8D 5180              CON(5)  #815
      0
E7B92 00                CON(2)  0       NO VARIABLES
                *
                *
E7B94 30                CON(2)  3       REGISTER LOGIC
E7B96 5                 CON(1)  5       OPCODE MNEMONIC
E7B97 44D3              NIBASC  \D=D!C \
      4412
      3402
E7BA3 50                CON(2)  (F.LSET)!(F.TREE)
E7BA5 4                 CON(1)  4       OPCODE
E7BA6 B0E0              CON(5)  #0E0B
      0
E7BAB 00                CON(2)  0       NO VARIABLES
                *
                *
E7BAD 50                CON(2)  5       FIXED
E7BAF 6                 CON(1)  6       OPCODE MNEMONIC
E7BB0 2545              NIBASC  \RTNSXM\
      E435
      85D4
E7BBC 50                CON(2)  (F.LSET)!(F.TREE)
E7BBE 2                 CON(1)  2       OPCODE
E7BBF 0000              CON(5)  #00
      0
E7BC4 00                CON(2)  0       NO VARIABLES
                *
                *
E7BC6 20                CON(2)  2       REGISTER ARITHMETIC
E7BC8 3                 CON(1)  3       OPCODE MNEMONIC
E7BC9 14D3              NIBASC  \A=B   \
      2402
      0202
E7BD5 40                CON(2)  (F.TREE)
E7BD7 3                 CON(1)  3       OPCODE
E7BD8 4000              CON(5)  #004
      0
E7BDD 2                 CON(1)  2       GROUP D
E7BDE 0                 CON(1)  0
                *
                *
E7BDF 00                CON(2)  0       NULL
E7BE1 0                 CON(1)  0       OPCODE MNEMONIC
E7BE2 0202              NIBASC  \      \
      0202
      0202
E7BEE 40                CON(2)  (F.TREE)
E7BF0 0                 CON(1)  0       OPCODE
E7BF1 0000              CON(5)  #0
      0
E7BF6 00                CON(2)  0       NO VARIABLES
```

```
753                     STITLE  HASH TO GROUP 7
754 E7BF8       OPCG07
755 E7BF8 50            CON(2)  5       FIXED
756 E7BFA 4             CON(1)  4       OPCODE MNEMONIC
757 E7BFB 14D3          NIBASC  \A=IN   \
          94E4
          0202
758 E7C07 10            CON(2)  (F.LSET)
759 E7C09 3             CON(1)  3       OPCODE
760 E7C0A 2080          CON(5)  #802
          0
761 E7C0F 00            CON(2)  0       NO VARIABLES
762             *
763             *
764 E7C11 90            CON(2)  9       STATUS TESTS
765 E7C13 5             CON(1)  5       OPCODE MNEMONIC
766 E7C14 F335          NIBASC  \?ST=0 \
          45D3
          0302
767 E7C20 32            CON(2)  (F.LSET)!(F.RETY)!(F.EXPR)
768 E7C22 3             CON(1)  3       OPCODE
769 E7C23 0680          CON(5)  #860
          0
770 E7C28 00            CON(2)  0       NO VARIABLES
771             *
772             *
773 E7C2A 40            CON(2)  4       BRANCHES
774 E7C2C 4             CON(1)  4       OPCODE MNEMONIC
775 E7C2D 74F4          NIBASC  \GOTO   \
          45F4
          0202
776 E7C39 12            CON(2)  (F.LSET)!(F.EXPR)
777 E7C3B 4             CON(1)  4       OPCODE
778 E7C3C 0006          CON(5)  #6000
          0
779 E7C41 2             CON(1)  2       FILL.MARK
780 E7C42 3             CON(1)  3       FILL.LENGTH
781             *
782             *
783 E7C43 10            CON(2)  1       REGISTER TESTS
784 E7C45 5             CON(1)  5       OPCODE MNEMONIC
785 E7C46 F334          NIBASC  \?C>=A \
          E3D3
          1402
786 E7C52 70            CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
787 E7C54 3             CON(1)  3       OPCODE
788 E7C55 A080          CON(5)  #80A
          0
789 E7C5A 2             CON(1)  2       GROUP B
790 E7C5B 0             CON(1)  0
791             *
792             *
793 E7C5C 30            CON(2)  3       REGISTER LOGIC
794 E7C5E 5             CON(1)  5       OPCODE MNEMONIC
795 E7C5F 14D3          NIBASC  \A=C&A \
```



```
          3462
          1402
796 E7C6B 50            CON(2)  (F.LSET)!(F.TREE)
797 E7C6D 4             CON(1)  4       OPCODE
798 E7C6E 60E0          CON(5)  #0E06
          0
799 E7C73 00            CON(2)  0       NO VARIABLES
800             *
801             *
802 E7C75 20            CON(2)  2       REGISTER ARITHMETIC
803 E7C77 5             CON(1)  5       OPCODE MNEMONIC
804 E7C78 34D3          NIBASC  \C=C+C \
          34B2
          3402
805 E7C84 40            CON(2)  (F.TREE)
806 E7C86 3             CON(1)  3       OPCODE
807 E7C87 6000          CON(5)  #006
          0
808 E7C8C 1             CON(1)  1       GROUP C
809 E7C8D 0             CON(1)  0
810             *
811             *
812 E7C8E 50            CON(2)  5       FIXED
813 E7C90 4             CON(1)  4       OPCODE MNEMONIC
814 E7C91 3545          NIBASC  \ST=C   \
          D334
          0202
815 E7C9D 50            CON(2)  (F.LSET)!(F.TREE)
816 E7C9F 2             CON(1)  2       OPCODE
817 E7CA0 A000          CON(5)  #0A
          0
818 E7CA5 00            CON(2)  0       NO VARIABLES
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER

```
819                     STITLE  HASH TO GROUP 8
820 E7CA7       OPCG08
821 E7CA7 50            CON(2)  5       FIXED
822 E7CA9 4             CON(1)  4       OPCODE MNEMONIC
823 E7CAA 3435          NIBASC  \CSRC  \
          2534
          0202
824 E7CB6 10            CON(2)  (F.LSET)
825 E7CB8 3             CON(1)  3       OPCODE
826 E7CB9 6180          CON(5)  #816
          0
827 E7CBE 00            CON(2)  0       NO VARIABLES
828             *
829             *
830 E7CC0 20            CON(2)  2       REGISTER ARITHMETIC
831 E7CC2 5             CON(1)  5       OPCODE MNEMONIC
832 E7CC3 24D3          NIBASC  \B=B-C \
          24D2
          3402
833 E7CCF 00            CON(2)  0
834 E7CD1 3             CON(1)  3       OPCODE
835 E7CD2 1000          CON(5)  #001
          0
836 E7CD7 3             CON(1)  3       GROUP E
837 E7CD8 0             CON(1)  0
838             *
839             *
840 E7CD9 50            CON(2)  5       FIXED
841 E7CDB 4             CON(1)  4       OPCODE MNEMONIC
842 E7CDC 3524          NIBASC  \SB=0  \
          D303
          0202
843 E7CE8 10            CON(2)  (F.LSET)
844 E7CEA 3             CON(1)  3       OPCODE
845 E7CEB 2280          CON(5)  #822
          0
846 E7CF0 00            CON(2)  0       NO VARIABLES
847             *
848             *
849 E7CF2 90            CON(2)  9       STATUS TESTS
850 E7CF4 5             CON(1)  5       OPCODE MNEMONIC
851 E7CF5 F335          NIBASC  \?ST#0 \
          4532
          0302
852 E7D01 32            CON(2)  (F.LSET)!(F.RETY)!(F.EXPR)
853 E7D03 3             CON(1)  3       OPCODE
854 E7D04 0780          CON(5)  #870
          0
855 E7D09 00            CON(2)  0       NO VARIABLES
856             *
857             *
858 E7D0B 20            CON(2)  2       REGISTER ARITHMETIC
859 E7D0D 4             CON(1)  4       OPCODE MNEMONIC
860 E7D0E 3424          NIBASC  \CBEX  \
          5485
```



```
          0202
861 E7D1A 00            CON(2)  0
862 E7D1C 3             CON(1)  3       OPCODE
863 E7D1D D000          CON(5)  #00D
          0
864 E7D22 2             CON(1)  2       GROUP D
865 E7D23 0             CON(1)  0
866             *
867             *
868 E7D24 C0            CON(2)  12      DATA POINTER ARITHMETIC
869 E7D26 6             CON(1)  6       OPCODE MNEMONIC
870 E7D27 4413          NIBASC  \D1=D1+\
          D344
          13B2
871 E7D33 12            CON(2)  (F.LSET)!(F.EXPR)
872 E7D35 3             CON(1)  3       OPCODE
873 E7D36 0710          CON(5)  #170
          0
874 E7D3B 00            CON(2)  0       NO VARIABLES
875             *
876             *
877 E7D3D B0            CON(2)  11      SET STATUS
878 E7D3F 4             CON(1)  4       OPCODE MNEMONIC
879 E7D40 3545          NIBASC  \ST=1  \
          D313
          0202
880 E7D4C 12            CON(2)  (F.LSET)!(F.EXPR)
881 E7D4E 3             CON(1)  3       OPCODE
882 E7D4F 0580          CON(5)  #850
          0
883 E7D54 00            CON(2)  0       NO VARIABLES
884             *
885             *
886 E7D56 10            CON(2)  1       REGISTER TESTS
887 E7D58 4             CON(1)  4       OPCODE MNEMONIC
888 E7D59 F314          NIBASC  \?A<B  \
          C324
          0202
889 E7D65 70            CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
890 E7D67 3             CON(1)  3       OPCODE
891 E7D68 4080          CON(5)  #804
          0
892 E7D6D 2             CON(1)  2       GROUP B
893 E7D6E 0             CON(1)  0
894             *
895             *
896 E7D6F D0            CON(2)  13      DATA TRANSFER
897 E7D71 6             CON(1)  6       OPCODE MNEMONIC
898 E7D72 14D3          NIBASC  \A=DAT0\
          4414
          4503
899 E7D7E 42            CON(2)  (F.TREE)!(F.EXPR)
900 E7D80 4             CON(1)  4       OPCODE
901 E7D81 0251          CON(5)  #1520
          0
```

```
902 E7D86 00          CON(2)  0       NO VARIABLES
903             *
904             *
905 E7D88 50          CON(2)  5       FIXED
906 E7D8A 4           CON(1)  4       OPCODE MNEMONIC
907 E7D8B 34D3        NIBASC  \C=R4  \
          2543
          0202
908 E7D97 50          CON(2)  (F.LSET)!(F.TREE)
909 E7D99 3           CON(1)  3       OPCODE
910 E7D9A C110        CON(5)  #11C
          0
911 E7D9F 00          CON(2)  0       NO VARIABLES
912             *
913             *
914 E7DA1 50          CON(2)  5       FIXED
915 E7DA3 5           CON(1)  5       OPCODE MNEMONIC
916 E7DA4 34C4        NIBASC  \CLRST \
          2535
          4502
917 E7DB0 50          CON(2)  (F.LSET)!(F.TREE)
918 E7DB2 2           CON(1)  2       OPCODE
919 E7DB3 8000        CON(5)  #08
          0
920 E7DB8 00          CON(2)  0       NO VARIABLES
921             *
922             *
923 E7DBA 50          CON(2)  5       FIXED
924 E7DBC 4           CON(1)  4       OPCODE MNEMONIC
925 E7DBD 4403        NIBASC  \D0=C  \
          D334
          0202
926 E7DC9 50          CON(2)  (F.LSET)!(F.TREE)
927 E7DCB 3           CON(1)  3       OPCODE
928 E7DCC 4310        CON(5)  #134
          0
929 E7DD1 00          CON(2)  0       NO VARIABLES
930             *
931             *
932 E7DD3 20          CON(2)  2       REGISTER ARITHMETIC
933 E7DD5 3           CON(1)  3       OPCODE MNEMONIC
934 E7DD6 44D3        NIBASC  \D=C   \
          3402
          0202
935 E7DE2 40          CON(2)  (F.TREE)
936 E7DE4 3           CON(1)  3       OPCODE
937 E7DE5 7000        CON(5)  #007
          0
938 E7DEA 2           CON(1)  2       GROUP D
939 E7DEB 0           CON(1)  0
940             *
941             *
942 E7DEC 50          CON(2)  5       FIXED
943 E7DEE 4           CON(1)  4       OPCODE MNEMONIC
944 E7DEF 3525        NIBASC  \SR=0  \
```



```
          D303
          0202
945 E7DFB 50          CON(2)  (F.LSET)!(F.TREE)
946 E7DFD 3           CON(1)  3       OPCODE
947 E7DFE 4280        CON(5)  #824
          0
948 E7E03 00          CON(2)  0       NO VARIABLES
949             *
950             *
951 E7E05 00          CON(2)  0       NULL
952 E7E07 0           CON(1)  0       OPCODE MNEMONIC
953 E7E08 0202        NIBASC  \      \
          0202
          0202
954 E7E14 40          CON(2)  (F.TREE)
955 E7E16 0           CON(1)  0       OPCODE
956 E7E17 0000        CON(5)  #0
          0
957 E7E1C 00          CON(2)  0       NO VARIABLES
```

```
 958                    STITLE  HASH TO GROUP 9
 959 E7E1E      OPCG09
 960 E7E1E 20           CON(2)  2       REGISTER ARITHMETIC
 961 E7E20 5            CON(1)  5       OPCODE MNEMONIC
 962 E7E21 24D3         NIBASC  \B=B+C \
           24B2
           3402
 963 E7E2D 00           CON(2)  0
 964 E7E2F 3            CON(1)  3       OPCODE
 965 E7E30 1000         CON(5)  #001
           0
 966 E7E35 1            CON(1)  1       GROUP C
 967 E7E36 0            CON(1)  0
 968             *
 969             *
 970 E7E37 10           CON(2)  1       REGISTER TESTS
 971 E7E39 4            CON(1)  4       OPCODE MNEMONIC
 972 E7E3A F334         NIBASC  \?C<A  \
           C314
           0202
 973 E7E46 30           CON(2)  (F.LSET)!(F.RETY)
 974 E7E48 3            CON(1)  3       OPCODE
 975 E7E49 6080         CON(5)  #806
           0
 976 E7E4E 2            CON(1)  2       GROUP B
 977 E7E4F 0            CON(1)  0
 978             *
 979             *
 980 E7E50 50           CON(2)  5       FIXED
 981 E7E52 4            CON(1)  4       OPCODE MNEMONIC
 982 E7E53 4403         NIBASC  \D0=A  \
           D314
           0202
 983 E7E5F 10           CON(2)  (F.LSET)
 984 E7E61 3            CON(1)  3       OPCODE
 985 E7E62 0310         CON(5)  #130
           0
 986 E7E67 00           CON(2)  0       NO VARIABLES
 987             *
 988             *
 989 E7E69 10           CON(2)  1       REGISTER TESTS
 990 E7E6B 5            CON(1)  5       OPCODE MNEMONIC
 991 E7E6C F314         NIBASC  \?A<=C \
           C3D3
           3402
 992 E7E78 70           CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
 993 E7E7A 3            CON(1)  3       OPCODE
 994 E7E7B A080         CON(5)  #80A
           0
 995 E7E80 2            CON(1)  2       GROUP B
 996 E7E81 0            CON(1)  0
 997             *
 998             *
 999 E7E82 D0           CON(2)  13      DATA TRANSFER
1000 E7E84 6            CON(1)  6       OPCODE MNEMONIC
```



```
1001 E7E85 14D3         NIBASC  \A=DAT1\
           4414
           4513
1002 E7E91 42           CON(2)  (F.TREE)!(F.EXPR)
1003 E7E93 4            CON(1)  4       OPCODE
1004 E7E94 0351         CON(5)  #1530
           0
1005 E7E99 00           CON(2)  0       NO VARIABLES
1006            *
1007            *
1008 E7E9B 50           CON(2)  5       FIXED
1009 E7E9D 4            CON(1)  4       OPCODE MNEMONIC
1010 E7E9E 34D3         NIBASC  \C=R2  \
           2523
           0202
1011 E7EAA 50           CON(2)  (F.LSET)!(F.TREE)
1012 E7EAC 3            CON(1)  3       OPCODE
1013 E7EAD A110         CON(5)  #11A
           0
1014 E7EB2 00           CON(2)  0       NO VARIABLES
1015            *
1016            *
1017 E7EB4 40           CON(2)  4       BRANCHES
1018 E7EB6 6            CON(1)  6       OPCODE MNEMONIC
1019 E7EB7 2554         NIBASC  \REL(1)\
           C482
           1392
1020 E7EC3 52           CON(2)  (F.LSET)!(F.TREE)!(F.EXPR)
1021 E7EC5 1            CON(1)  1       OPCODE
1022 E7EC6 0000         CON(5)  #0
           0
1023 E7ECB 1            CON(1)  1       FILL.MARK
1024 E7ECC 1            CON(1)  1       FILL.LENGTH
```

```
1025                    STITLE  HASH TO GROUP 10
1026 E7ECD      OPCG10
1027 E7ECD 50           CON(2)  5       FIXED
1028 E7ECF 4            CON(1)  4       OPCODE MNEMONIC
1029 E7ED0 34D3         NIBASC  \C=R0  \
           2503
           0202
1030 E7EDC 10           CON(2)  (F.LSET)
1031 E7EDE 3            CON(1)  3       OPCODE
1032 E7EDF 8110         CON(5)  #118
           0
1033 E7EE4 00           CON(2)  0       NO VARIABLES
1034            *
1035            *
1036 E7EE6 30           CON(2)  3       REGISTER LOGIC
1037 E7EE8 5            CON(1)  5       OPCODE MNEMONIC
1038 E7EE9 24D3         NIBASC  \B=B!C \
           2412
           3402
1039 E7EF5 10           CON(2)  (F.LSET)
1040 E7EF7 4            CON(1)  4       OPCODE
1041 E7EF8 90E0         CON(5)  #0E09
           0
1042 E7EFD 00           CON(2)  0       NO VARIABLES
1043            *
1044            *
1045 E7EFF 50           CON(2)  5       FIXED
1046 E7F01 4            CON(1)  4       OPCODE MNEMONIC
1047 E7F02 4435         NIBASC  \DSRC  \
           2534
           0202
1048 E7F0E 10           CON(2)  (F.LSET)
1049 E7F10 3            CON(1)  3       OPCODE
1050 E7F11 7180         CON(5)  #817
           0
1051 E7F16 00           CON(2)  0       NO VARIABLES
1052            *
1053            *
1054 E7F18 20           CON(2)  2       REGISTER ARITHMETIC
1055 E7F1A 5            CON(1)  5       OPCODE MNEMONIC
1056 E7F1B 14D3         NIBASC  \A=A-C \
           14D2
           3402
1057 E7F27 40           CON(2)  (F.TREE)
1058 E7F29 3            CON(1)  3       OPCODE
1059 E7F2A A000         CON(5)  #00A
           0
1060 E7F2F 3            CON(1)  3       GROUP E
1061 E7F30 0            CON(1)  0
1062            *
1063            *
1064 E7F31 20           CON(2)  2       REGISTER ARITHMETIC
1065 E7F33 3            CON(1)  3       OPCODE MNEMONIC
1066 E7F34 34D3         NIBASC  \C=B   \
           2402
```



```
           0202
1067 E7F40 40           CON(2)  (F.TREE)
1068 E7F42 3            CON(1)  3       OPCODE
1069 E7F43 9000         CON(5)  #009
           0
1070 E7F48 2            CON(1)  2       GROUP D
1071 E7F49 0            CON(1)  0
1072            *
1073            *
1074 E7F4A C0           CON(2)  12      DATA POINTER ARITHMETIC
1075 E7F4C 6            CON(1)  6       OPCODE MNEMONIC
1076 E7F4D 4413         NIBASC  \D1=D1-\
           D344
           13D2
1077 E7F59 52           CON(2)  (F.LSET)!(F.EXPR)!(F.TREE)
1078 E7F5B 3            CON(1)  3       OPCODE
1079 E7F5C 0C10         CON(5)  #1C0
           0
1080 E7F61 00           CON(2)  0       NO VARIABLES
1081            *
1082            *
1083 E7F63 50           CON(2)  5       FIXED
1084 E7F65 5            CON(1)  5       OPCODE MNEMONIC
1085 E7F66 F455         NIBASC  \OUT=C \
           45D3
           3402
1086 E7F72 50           CON(2)  (F.LSET)!(F.TREE)
1087 E7F74 3            CON(1)  3       OPCODE
1088 E7F75 1080         CON(5)  #801
           0
1089 E7F7A 00           CON(2)  0       NO VARIABLES
```

```
1090                    STITLE  HASH TO GROUP 11
1091 E7F7C      OPCG11
1092 E7F7C 20           CON(2)  2       REGISTER ARITHMETIC
1093 E7F7E 5            CON(1)  5       OPCODE MNEMONIC
1094 E7F7F 34D3         NIBASC  \C=B+C \
          24B2
          3402
1095 E7F8B 00           CON(2)  0
1096 E7F8D 3            CON(1)  3       OPCODE
1097 E7F8E 9000         CON(5)  #009
          0
1098 E7F93 1            CON(1)  1       GROUP C
1099 E7F94 0            CON(1)  0
1100            *
1101            *
1102 E7F95 20           CON(2)  2       REGISTER ARITHMETIC
1103 E7F97 5            CON(1)  5       OPCODE MNEMONIC
1104 E7F98 14D3         NIBASC  \A=A+C \
          14B2
          3402
1105 E7FA4 00           CON(2)  0
1106 E7FA6 3            CON(1)  3       OPCODE
1107 E7FA7 A000         CON(5)  #00A
          0
1108 E7FAC 1            CON(1)  1       GROUP C
1109 E7FAD 0            CON(1)  0
1110            *
1111            *
1112 E7FAE 50           CON(2)  5       FIXED
1113 E7FB0 4            CON(1)  4       OPCODE MNEMONIC
1114 E7FB1 34D3         NIBASC  \C=IN  \
          94E4
          0202
1115 E7FBD 10           CON(2)  (F.LSET)
1116 E7FBF 3            CON(1)  3       OPCODE
1117 E7FC0 3080         CON(5)  #803
          0
1118 E7FC5 00           CON(2)  0       NO VARIABLES
1119            *
1120            *
1121 E7FC7 10           CON(2)  1       REGISTER TESTS
1122 E7FC9 4            CON(1)  4       OPCODE MNEMONIC
1123 E7FCA F344         NIBASC  \?D>C  \
          E334
          0202
1124 E7FD6 70           CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
1125 E7FD8 3            CON(1)  3       OPCODE
1126 E7FD9 3080         CON(5)  #803
          0
1127 E7FDE 2            CON(1)  2       GROUP B
1128 E7FDF 0            CON(1)  0
1129            *
1130            *
1131 E7FE0 30           CON(2)  3       REGISTER LOGIC
1132 E7FE2 5            CON(1)  5       OPCODE MNEMONIC
```



```
1133 E7FE3 14D3         NIBASC  \A=B&A \
          2462
          1402
1134 E7FEF 50           CON(2)  (F.LSET)!(F.TREE)
1135 E7FF1 4            CON(1)  4       OPCODE
1136 E7FF2 00E0         CON(5)  #0E00
          0
1137 E7FF7 00           CON(2)  0       NO VARIABLES
1138            *
1139            *
1140 E7FF9 30           CON(2)  3       REGISTER LOGIC
1141 E7FFB 5            CON(1)  5       OPCODE MNEMONIC
1142 E7FFC 34D3         NIBASC  \C=C&A \
          3462
          1402
1143 E8008 50           CON(2)  (F.LSET)!(F.TREE)
1144 E800A 4            CON(1)  4       OPCODE
1145 E800B 20E0         CON(5)  #0E02
          0
1146 E8010 00           CON(2)  0       NO VARIABLES
1147            *
1148            *
1149 E8012 32           CON(2)  35      PSEUDO-OP
1150 E8014 5            CON(1)  5       OPCODE MNEMONIC
1151 E8015 45F4         NIBASC  \TOKEN \
          B454
          E402
1152 E8021 42           CON(2)  (F.TREE)!(F.EXPR)
1153 E8023 2            CON(1)  2       OPCODE
1154 E8024 0000         CON(5)  #0
          0
1155 E8029 00           CON(2)  0       NO VARIABLES
```

```
1156                    STITLE  HASH TO GROUP 12
1157 E802B      OPCG12
1158 E802B 50           CON(2)  5       FIXED
1159 E802D 4            CON(1)  4       OPCODE MNEMONIC
1160 E802E 34D3         NIBASC  \C=ID  \
          9444
          0202
1161 E803A 10           CON(2)  (F.LSET)
1162 E803C 3            CON(1)  3       OPCODE
1163 E803D 6080         CON(5)  #806
          0
1164 E8042 00           CON(2)  0       NO VARIABLES
1165            *
1166            *
1167 E8044 20           CON(2)  2       REGISTER ARITHMETIC
1168 E8046 3            CON(1)  3       OPCODE MNEMONIC
1169 E8047 24D3         NIBASC  \B=A   \
          1402
          0202
1170 E8053 00           CON(2)  0
1171 E8055 3            CON(1)  3       OPCODE
1172 E8056 8000         CON(5)  #008
          0
1173 E805B 2            CON(1)  2       GROUP D
1174 E805C 0            CON(1)  0
1175            *
1176            *
1177 E805D A0           CON(2)  10      SET POINTER
1178 E805F 2            CON(1)  2       OPCODE MNEMONIC
1179 E8060 05D3         NIBASC  \P=    \
          0202
          0202
1180 E806C 12           CON(2)  (F.LSET)!(F.EXPR)
1181 E806E 2            CON(1)  2       OPCODE
1182 E806F 0200         CON(5)  #20
          0
1183 E8074 00           CON(2)  0       NO VARIABLES
1184            *
1185            *
1186 E8076 10           CON(2)  1       REGISTER TESTS
1187 E8078 5            CON(1)  5       OPCODE MNEMONIC
1188 E8079 F324         NIBASC  \?B>=C \
          E3D3
          3402
1189 E8085 30           CON(2)  (F.LSET)!(F.RETY)
1190 E8087 3            CON(1)  3       OPCODE
1191 E8088 9080         CON(5)  #809
          0
1192 E808D 2            CON(1)  2       GROUP B
1193 E808E 0            CON(1)  0
1194            *
1195            *
1196 E808F D0           CON(2)  13      DATA TRANSFER
1197 E8091 6            CON(1)  6       OPCODE MNEMONIC
1198 E8092 34D3         NIBASC  \C=DAT0\
```



```
          4414
          4503
1199 E809E 02           CON(2)  (F.EXPR)
1200 E80A0 4            CON(1)  4       OPCODE
1201 E80A1 0651         CON(5)  #1560
          0
1202 E80A6 00           CON(2)  0       NO VARIABLES
1203            *
1204            *
1205 E80A8 E0           CON(2)  14      HEX CONSTANTS
1206 E80AA 6            CON(1)  6       OPCODE MNEMONIC
1207 E80AB E494         NIBASC  \NIBHEX\
          2484
          5485
1208 E80B7 40           CON(2)  (F.TREE)
1209 E80B9 0            CON(1)  0       OPCODE
1210 E80BA 0000         CON(5)  #0
          0
1211 E80BF 1            CON(1)  1       FILL.MARK
1212 E80C0 E            CON(1)  14      FILL.LENGTH
1213            *
1214            *
1215 E80C1 50           CON(2)  5       FIXED
1216 E80C3 5            CON(1)  5       OPCODE MNEMONIC
1217 E80C4 2554         NIBASC  \RESET \
          3554
          4502
1218 E80D0 50           CON(2)  (F.LSET)!(F.TREE)
1219 E80D2 3            CON(1)  3       OPCODE
1220 E80D3 A080         CON(5)  #80A
          0
1221 E80D8 00           CON(2)  0       NO VARIABLES
1222            *
1223            *
1224 E80DA 10           CON(2)  1       REGISTER TESTS
1225 E80DC 4            CON(1)  4       OPCODE MNEMONIC
1226 E80DD F314         NIBASC  \?A#B  \
          3224
          0202
1227 E80E9 70           CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
1228 E80EB 3            CON(1)  3       OPCODE
1229 E80EC 4080         CON(5)  #804
          0
1230 E80F1 1            CON(1)  1       GROUP A
1231 E80F2 0            CON(1)  0
1232            *
1233            *
1234 E80F3 30           CON(2)  3       REGISTER LOGIC
1235 E80F5 5            CON(1)  5       OPCODE MNEMONIC
1236 E80F6 14D3         NIBASC  \A=A!C \
          1412
          3402
1237 E8102 50           CON(2)  (F.LSET)!(F.TREE)
1238 E8104 4            CON(1)  4       OPCODE
1239 E8105 E0E0         CON(5)  #0E0E
```

```
          0
1240 E810A 00          CON(2)  0       NO VARIABLES
1241           *
1242           *
1243 E810C 30          CON(2)  3       REGISTER LOGIC
1244 E810E 5           CON(1)  5       OPCODE MNEMONIC
1245 E810F 34D3        NIBASC  \C=B!C \
          2412
          3402
1246 E811B 50          CON(2)  (F.LSET)!(F.TREE)
1247 E811D 4           CON(1)  4       OPCODE
1248 E811E D0E0        CON(5)  #0E0D
          0
1249 E8123 00          CON(2)  0       NO VARIABLES
1250           *
1251           *
1252 E8125 00          CON(2)  0       NULL
1253 E8127 0           CON(1)  0       OPCODE MNEMONIC
1254 E8128 0202        NIBASC  \      \
          0202
          0202
1255 E8134 40          CON(2)  (F.TREE)
1256 E8136 0           CON(1)  0       OPCODE
1257 E8137 0000        CON(5)  #0
          0
1258 E813C 00          CON(2)  0       NO VARIABLES
```



```
1259                   STITLE  HASH TO GROUP 13
1260 E813E      OPCG13
1261 E813E 30          CON(2)  3       REGISTER LOGIC
1262 E8140 5           CON(1)  5       OPCODE MNEMONIC
1263 E8141 24D3        NIBASC  \B=B&A \
          2462
          1402
1264 E814D 10          CON(2)  (F.LSET)
1265 E814F 4           CON(1)  4       OPCODE
1266 E8150 40E0        CON(5)  #0E04
          0
1267 E8155 00          CON(2)  0       NO VARIABLES
1268           *
1269           *
1270 E8157 60          CON(2)  6       HARDWARE TESTS
1271 E8159 5           CON(1)  5       OPCODE MNEMONIC
1272 E815A F385        NIBASC  \?XM=0 \
          D4D3
          0302
1273 E8166 30          CON(2)  (F.LSET)!(F.RETY)
1274 E8168 3           CON(1)  3       OPCODE
1275 E8169 1380        CON(5)  #831
          0
1276 E816E 00          CON(2)  0       NO VARIABLES
1277           *
1278           *
1279 E8170 20          CON(2)  2       REGISTER ARITHMETIC
1280 E8172 5           CON(1)  5       OPCODE MNEMONIC
1281 E8173 44D3        NIBASC  \D=D+D \
          44B2
          4402
1282 E817F 00          CON(2)  0
1283 E8181 3           CON(1)  3       OPCODE
1284 E8182 7000        CON(5)  #007
          0
1285 E8187 1           CON(1)  1       GROUP C
1286 E8188 0           CON(1)  0
1287           *
1288           *
1289 E8189 10          CON(2)  1       REGISTER TESTS
1290 E818B 4           CON(1)  4       OPCODE MNEMONIC
1291 E818C F334        NIBASC  \?C#A  \
          3214
          0202
1292 E8198 30          CON(2)  (F.LSET)!(F.RETY)
1293 E819A 3           CON(1)  3       OPCODE
1294 E819B 6080        CON(5)  #806
          0
1295 E81A0 1           CON(1)  1       GROUP A
1296 E81A1 0           CON(1)  0
1297           *
1298           *
1299 E81A2 20          CON(2)  2       REGISTER ARITHMETIC
1300 E81A4 6           CON(1)  6       OPCODE MNEMONIC
1301 E81A5 14D3        NIBASC  \A=-A-1\
```

```
           D214
           D213
1302 E81B1 40          CON(2)  (F.TREE)
1303 E81B3 3           CON(1)  3       OPCODE
1304 E81B4 C000        CON(5)  #00C
           0
1305 E81B9 4           CON(1)  4       GROUP F
1306 E81BA 0           CON(1)  0
1307            *
1308            *
1309 E81BB D0          CON(2)  13      DATA TRANSFER
1310 E81BD 6           CON(1)  6       OPCODE MNEMONIC
1311 E81BE 34D3        NIBASC  \C=DAT1\
           4414
           4513
1312 E81CA 42          CON(2)  (F.EXPR)!(F.TREE)
1313 E81CC 4           CON(1)  4       OPCODE
1314 E81CD 0751        CON(5)  #1570
           0
1315 E81D2 00          CON(2)  0       NO VARIABLES
1316            *
1317            *
1318 E81D4 50          CON(2)  5       FIXED
1319 E81D6 6           CON(1)  6       OPCODE MNEMONIC
1320 E81D7 F455        NIBASC  \OUT=CS\
           45D3
           3435
1321 E81E3 50          CON(2)  (F.LSET)!(F.TREE)
1322 E81E5 3           CON(1)  3       OPCODE
1323 E81E6 0080        CON(5)  #800
           0
1324 E81EB 00          CON(2)  0       NO VARIABLES
1325            *
1326            *
1327 E81ED 10          CON(2)  1       REGISTER TESTS
1328 E81EF 4           CON(1)  4       OPCODE MNEMONIC
1329 E81F0 F314        NIBASC  \?A#0  \
           3203
           0202
1330 E81FC 70          CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
1331 E81FE 3           CON(1)  3       OPCODE
1332 E81FF C080        CON(5)  #80C
           0
1333 E8204 1           CON(1)  1       GROUP 1
1334 E8205 0           CON(1)  0
1335            *
1336            *
1337 E8206 00          CON(2)  0       NULL
1338 E8208 0           CON(1)  0       OPCODE MNEUMONIC
1339 E8209 0202        NIBASC  \      \
           0202
           0202
1340 E8215 40          CON(2)  (F.TREE)
1341 E8217 0           CON(1)  0       OPCODE
1342 E8218 0000        CON(5)  #0
```



```
           0
1343 E821D 00          CON(2)  0       NO VARIABLES
```

```
1344                    STITLE  HASH TO GROUP 14
1345 E821F      OPCG14
1346 E821F 20           CON(2)  2       REGISTER ARITHMETIC
1347 E8221 5            CON(1)  5       OPCODE MNEMONIC
1348 E8222 34D3         NIBASC  \C=C-D \
           34D2
           4402
1349 E822E 00           CON(2)  0
1350 E8230 3            CON(1)  3       OPCODE
1351 E8231 B000         CON(5)  #00B
           0
1352 E8236 3            CON(1)  3       GROUP E
1353 E8237 0            CON(1)  0
1354            *
1355            *
1356 E8238 20           CON(2)  2       REGISTER ARITHMETIC
1357 E823A 3            CON(1)  3       OPCODE MNEMONIC
1358 E823B 34D3         NIBASC  \C=A    \
           1402
           0202
1359 E8247 00           CON(2)  0
1360 E8249 3            CON(1)  3       OPCODE
1361 E824A 6000         CON(5)  #006
           0
1362 E824F 2            CON(1)  2       GROUP D
1363 E8250 0            CON(1)  0
1364            *
1365            *
1366 E8251 22           CON(2)  34      PSEUDO-OP
1367 E8253 3            CON(1)  3       OPCODE MNEMONIC
1368 E8254 B454         NIBASC  \KEY    \
           9502
           0202
1369 E8260 02           CON(2)  (F.EXPR)
1370 E8262 0            CON(1)  0       OPCODE
1371 E8263 0000         CON(5)  #0
           0
1372 E8268 00           CON(2)  0       NO VARIABLES
1373            *
1374            *
1375 E826A 20           CON(2)  2       REGISTER ARITHMETIC
1376 E826C 3            CON(1)  3       OPCODE MNEMONIC
1377 E826D 14D3         NIBASC  \A=0    \
           0302
           0202
1378 E8279 40           CON(2)  (F.TREE)
1379 E827B 3            CON(1)  3       OPCODE
1380 E827C 0000         CON(5)  #000
           0
1381 E8281 2            CON(1)  2       GROUP D
1382 E8282 0            CON(1)  0
1383            *
1384            *
1385 E8283 20           CON(2)  2       REGISTER ARITHMETIC
1386 E8285 5            CON(1)  5       OPCODE MNEMONIC
```



```
1387 E8286 34D3         NIBASC  \C=A-C \
           14D2
           3402
1388 E8292 40           CON(2)  (F.TREE)
1389 E8294 3            CON(1)  3       OPCODE
1390 E8295 E000         CON(5)  #00E
           0
1391 E829A 3            CON(1)  3       GROUP E
1392 E829B 0            CON(1)  0
1393            *
1394            *
1395 E829C 40           CON(2)  4       BRANCHES
1396 E829E 6            CON(1)  6       OPCODE MNEMONIC
1397 E829F 34F4         NIBASC  \CON(5)\
           E482
           5392
1398 E82AB 53           CON(2)  (F.LSET)!(F.TREE)!(F.ABSL)!(F.EXPR)
1399 E82AD 5            CON(1)  5       OPCODE
1400 E82AE 0000         CON(5)  #00000
           0
1401 E82B3 1            CON(1)  1       FILL.MARK
1402 E82B4 5            CON(1)  5       FILL.LENGTH
1403            *
1404            *
1405 E82B5 00           CON(2)  0       NULL
1406 E82B7 0            CON(1)  0       OPCODE MNEMONIC
1407 E82B8 0202         NIBASC  \       \
           0202
           0202
1408 E82C4 40           CON(2)  (F.TREE)
1409 E82C6 0            CON(1)  0       OPCODE
1410 E82C7 0000         CON(5)  #0
           0
1411 E82CC 00           CON(2)  0       NO VARIABLES
```

```
1412                    STITLE  HASH TO GROUP 15
1413 E82CE      OPCG15
1414 E82CE 20           CON(2)  2       REGISTER ARITHMETIC
1415 E82D0 4            CON(1)  4       OPCODE MNEMONIC
1416 E82D1 1434         NIBASC  \ACEX  \
           5485
           0202
1417 E82DD 00           CON(2)  0
1418 E82DF 3            CON(1)  3       OPCODE
1419 E82E0 E000         CON(5)  #00E
           0
1420 E82E5 2            CON(1)  2       GROUP D
1421 E82E6 0            CON(1)  0
1422            *
1423            *
1424 E82E7 60           CON(2)  6       HARDWARE TESTS
1425 E82E9 5            CON(1)  5       OPCODE MNEMONIC
1426 E82EA F335         NIBASC  \?SB=0 \
           24D3
           0302
1427 E82F6 30           CON(2)  (F.LSET)!(F.RETY)
1428 E82F8 3            CON(1)  3       OPCODE
1429 E82F9 2380         CON(5)  #832
           0
1430 E82FE 00           CON(2)  0       NO VARIABLES
1431            *
1432            *
1433 E8300 E1           CON(2)  30      PSEUDO-OP
1434 E8302 3            CON(1)  3       OPCODE MNEMONIC
1435 E8303 D435         NIBASC  \MSG   \
           7402
           0202
1436 E830F 12           CON(2)  (F.LSET)!(F.EXPR)
1437 E8311 4            CON(1)  4       OPCODE
1438 E8312 0000         CON(5)  #0
           0
1439 E8317 00           CON(2)  0       NO VARIABLES
1440            *
1441            *
1442 E8319 10           CON(2)  1       REGISTER TESTS
1443 E831B 5            CON(1)  5       OPCODE MNEMONIC
1444 E831C F334         NIBASC  \?C>=B \
           E3D3
           2402
1445 E8328 30           CON(2)  (F.LSET)!(F.RETY)
1446 E832A 3            CON(1)  3       OPCODE
1447 E832B D080         CON(5)  #80D
           0
1448 E8330 2            CON(1)  2       GROUP B
1449 E8331 0            CON(1)  0
1450            *
1451            *
1452 E8332 90           CON(2)  9       STATUS TESTS
1453 E8334 5            CON(1)  5       OPCODE MNEMONIC
1454 E8335 F335         NIBASC  \?ST=1 \
```



```
           45D3
           1302
1455 E8341 32           CON(2)  (F.LSET)!(F.RETY)!(F.EXPR)
1456 E8343 3            CON(1)  3       OPCODE
1457 E8344 0780         CON(5)  #870
           0
1458 E8349 00           CON(2)  0       NO VARIABLES
1459            *
1460            *
1461 E834B 20           CON(2)  2       REGISTER ARITHMETIC
1462 E834D 5            CON(1)  5       OPCODE MNEMONIC
1463 E834E 34D3         NIBASC  \C=C+D \
           34B2
           4402
1464 E835A 00           CON(2)  0
1465 E835C 3            CON(1)  3       OPCODE
1466 E835D B000         CON(5)  #00B
           0
1467 E8362 1            CON(1)  1       GROUP C
1468 E8363 0            CON(1)  0
1469            *
1470            *
1471 E8364 50           CON(2)  5       FIXED
1472 E8366 4            CON(1)  4       OPCODE MNEMONIC
1473 E8367 2513         NIBASC  \R1=C  \
           D334
           0202
1474 E8373 50           CON(2)  (F.LSET)!(F.TREE)
1475 E8375 3            CON(1)  3       OPCODE
1476 E8376 9010         CON(5)  #109
           0
1477 E837B 00           CON(2)  0       NO VARIABLES
1478            *
1479            *
1480 E837D 10           CON(2)  1       REGISTER TESTS
1481 E837F 5            CON(1)  5       OPCODE MNEMONIC
1482 E8380 F334         NIBASC  \?C<=A \
           C3D3
           1402
1483 E838C 70           CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
1484 E838E 3            CON(1)  3       OPCODE
1485 E838F E080         CON(5)  #80E
           0
1486 E8394 2            CON(1)  2       GROUP B
1487 E8395 0            CON(1)  0
1488            *
1489            *
1490 E8396 10           CON(2)  1       REGISTER TESTS
1491 E8398 4            CON(1)  4       OPCODE MNEMONIC
1492 E8399 F344         NIBASC  \?D=C  \
           D334
           0202
1493 E83A5 70           CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
1494 E83A7 3            CON(1)  3       OPCODE
1495 E83A8 3080         CON(5)  #803
```

```
          0
1496 E83AD 1          CON(1)  1       GROUP A
1497 E83AE 0          CON(1)  0
1498           *
1499           *
1500 E83AF 60         CON(2)  6       HARDWARE TESTS
1501 E83B1 5          CON(1)  5       OPCODE MNEMONIC
1502 E83B2 F335       NIBASC  \?SR=0 \
          25D3
          0302
1503 E83BE 70         CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
1504 E83C0 3          CON(1)  3       OPCODE
1505 E83C1 4380       CON(5)  #834
          0
1506 E83C6 00         CON(2)  0       NO VARIABLES
1507           *
1508           *
1509 E83C8 20         CON(2)  2       REGISTER ARITHMETIC
1510 E83CA 4          CON(1)  4       OPCODE MNEMONIC
1511 E83CB 14D3       NIBASC  \A=-A  \
          D214
          0202
1512 E83D7 40         CON(2)  (F.TREE)
1513 E83D9 3          CON(1)  3       OPCODE
1514 E83DA 8000       CON(5)  #008
          0
1515 E83DF 4          CON(1)  4       GROUP F
1516 E83E0 0          CON(1)  0
1517           *
1518           *
1519 E83E1 20         CON(2)  2       REGISTER ARITHMETIC
1520 E83E3 5          CON(1)  5       OPCODE MNEMONIC
1521 E83E4 34D3       NIBASC  \C=A+C \
          14B2
          3402
1522 E83F0 40         CON(2)  (F.TREE)
1523 E83F2 3          CON(1)  3       OPCODE
1524 E83F3 2000       CON(5)  #002
          0
1525 E83F8 1          CON(1)  1       GROUP C
1526 E83F9 0          CON(1)  0
1527           *
1528           *
1529 E83FA 00         CON(2)  0       NULL
1530 E83FC 0          CON(1)  0       OPCODE MNEMONIC
1531 E83FD 0202       NIBASC  \      \
          0202
          0202
1532 E8409 40         CON(2)  (F.TREE)
1533 E840B 0          CON(1)  0       OPCODE
1534 E840C 0000       CON(5)  #0
          0
1535 E8411 00         CON(2)  0       NO VARIABLES
```



```
1536                  STITLE  HASH TO GROUP 16
1537 E8413      OPCG16
1538 E8413 40         CON(2)  4       BRANCHES
1539 E8415 6          CON(1)  6       OPCODE MNEMONIC
1540 E8416 4413       NIBASC  \D1=(2)\
          D382
          2392
1541 E8422 1B         CON(2)  (F.LSET)!(F.ABSL)!(F.EXPR)!(F.PART)
1542 E8424 4          CON(1)  4       OPCODE
1543 E8425 00D1       CON(5)  #1D00
          0
1544 E842A 3          CON(1)  3       FILL.MARK
1545 E842B 2          CON(1)  2       FILL.LENGTH
1546           *
1547           *
1548 E842C 20         CON(2)  2       REGISTER ARITHMETIC
1549 E842E 6          CON(1)  6       OPCODE MNEMONIC
1550 E842F 34D3       NIBASC  \C=-C-1\
          D234
          D213
1551 E843B 00         CON(2)  0
1552 E843D 3          CON(1)  3       OPCODE
1553 E843E E000       CON(5)  #00E
          0
1554 E8443 4          CON(1)  4       GROUP F
1555 E8444 0          CON(1)  0
1556           *
1557           *
1558 E8445 F1         CON(2)  31      PSEUDO-OP
1559 E8447 4          CON(1)  4       OPCODE MNEMONIC
1560 E8448 05F4       NIBASC  \POLL  \
          C4C4
          0202
1561 E8454 12         CON(2)  (F.LSET)!(F.EXPR)
1562 E8456 5          CON(1)  5       OPCODE
1563 E8457 0000       CON(5)  #0
          0
1564 E845C 00         CON(2)  0       NO VARIABLES
1565           *
1566           *
1567 E845E 90         CON(2)  9       STATUS TESTS
1568 E8460 5          CON(1)  5       OPCODE MNEMONIC
1569 E8461 F335       NIBASC  \?ST#1 \
          4532
          1302
1570 E846D 32         CON(2)  (F.LSET)!(F.RETY)!(F.EXPR)
1571 E846F 3          CON(1)  3       OPCODE
1572 E8470 0680       CON(5)  #860
          0
1573 E8475 00         CON(2)  0       NO VARIABLES
1574           *
1575           *
1576 E8477 30         CON(2)  3       REGISTER LOGIC
1577 E8479 5          CON(1)  5       OPCODE MNEMONIC
1578 E847A 34D3       NIBASC  \C=C!D \
```

```
            3412
            4402
1579 E8486  10          CON(2)  (F.LSET)
1580 E8488  4           CON(1)  4       OPCODE
1581 E8489  F0E0        CON(5)  #0E0F
            0
1582 E848E  00          CON(2)  0       NO VARIABLES
1583            *
1584            *
1585 E8490  20          CON(2)  2       REGISTER ARITHMETIC
1586 E8492  5           CON(1)  5       OPCODE MNEMONIC
1587 E8493  44D3        NIBASC  \D=C-D \
            34D2
            4402
1588 E849F  40          CON(2)  (F.TREE)
1589 E84A1  3           CON(1)  3       OPCODE
1590 E84A2  F000        CON(5)  #00F
            0
1591 E84A7  3           CON(1)  3       GROUP E
1592 E84A8  0           CON(1)  0
1593            *
1594            *
1595 E84A9  50          CON(2)  5       FIXED
1596 E84AB  4           CON(1)  4       OPCODE MNEMONIC
1597 E84AC  2513        NIBASC  \R1=A  \
            D314
            0202
1598 E84B8  50          CON(2)  (F.LSET)!(F.TREE)
1599 E84BA  3           CON(1)  3       OPCODE
1600 E84BB  1010        CON(5)  #101
            0
1601 E84C0  00          CON(2)  0       NO VARIABLES
1602            *
1603            *
1604 E84C2  10          CON(2)  1       REGISTER TESTS
1605 E84C4  4           CON(1)  4       OPCODE MNEMONIC
1606 E84C5  F324        NIBASC  \?B=0  \
            D303
            0202
1607 E84D1  70          CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
1608 E84D3  3           CON(1)  3       OPCODE
1609 E84D4  9080        CON(5)  #809
            0
1610 E84D9  1           CON(1)  1       GROUP A
1611 E84DA  0           CON(1)  0
1612            *
1613            *
1614 E84DB  20          CON(2)  2       REGISTER ARITHMETIC
1615 E84DD  3           CON(1)  3       OPCODE MNEMONIC
1616 E84DE  24D3        NIBASC  \B=0   \
            0302
            0202
1617 E84EA  40          CON(2)  (F.TREE)
1618 E84EC  3           CON(1)  3       OPCODE
1619 E84ED  1000        CON(5)  #001
```



```
            0
1620 E84F2  2           CON(1)  2       GROUP D
1621 E84F3  0           CON(1)  0
1622            *
1623            *
1624 E84F4  30          CON(2)  3       REGISTER LOGIC
1625 E84F6  5           CON(1)  5       OPCODE MNEMONIC
1626 E84F7  34D3        NIBASC  \C=A!C \
            1412
            3402
1627 E8503  50          CON(2)  (F.LSET)!(F.TREE)
1628 E8505  4           CON(1)  4       OPCODE
1629 E8506  A0E0        CON(5)  #0E0A
            0
1630 E850B  00          CON(2)  0       NO VARIABLES
1631            *
1632            *
1633 E850D  00          CON(2)  0       NULL
1634 E850F  0           CON(1)  0       OPCODE MNEMONIC
1635 E8510  0202        NIBASC  \      \
            0202
            0202
1636 E851C  40          CON(2)  (F.TREE)
1637 E851E  0           CON(1)  0       OPCODE
1638 E851F  0000        CON(5)  #0
            0
1639 E8524  00          CON(2)  0       NO VARIABLES
```

```
1640                    STITLE  HASH TO GROUP 17
1641 E8526      OPCG17
1642 E8526 51           CON(2)  21      END PSEUDO-OP
1643 E8528 3            CON(1)  3       OPCODE MNEMONIC
1644 E8529 54E4         NIBASC  \END   \
           4402
           0202
1645 E8535 00           CON(2)  0
1646 E8537 0            CON(1)  0       OPCODE
1647 E8538 0000         CON(5)  #0
           0
1648 E853D 00           CON(2)  0       NO VARIABLES
1649            *
1650            *
1651 E853F 20           CON(2)  2       REGISTER ARITHMETIC
1652 E8541 4            CON(1)  4       OPCODE MNEMONIC
1653 E8542 2434         NIBASC  \BCEX  \
           5485
           0202
1654 E854E 00           CON(2)  0
1655 E8550 3            CON(1)  3       OPCODE
1656 E8551 D000         CON(5)  #00D
           0
1657 E8556 2            CON(1)  2       GROUP D
1658 E8557 0            CON(1)  0
1659            *
1660            *
1661 E8558 40           CON(2)  4       BRANCHES
1662 E855A 6            CON(1)  6       OPCODE MNEMONIC
1663 E855B 2554         NIBASC  \REL(2)\
           C482
           2392
1664 E8567 12           CON(2)  (F.LSET)!(F.EXPR)
1665 E8569 2            CON(1)  2       OPCODE
1666 E856A 0000         CON(5)  #0
           0
1667 E856F 1            CON(1)  1       FILL.MARK
1668 E8570 2            CON(1)  2       FILL.LENGTH
1669            *
1670            *
1671 E8571 30           CON(2)  3       REGISTER LOGIC
1672 E8573 5            CON(1)  5       OPCODE MNEMONIC
1673 E8574 24D3         NIBASC  \B=C&B \
           3462
           2402
1674 E8580 50           CON(2)  (F.LSET)!(F.TREE)
1675 E8582 4            CON(1)  4       OPCODE
1676 E8583 10E0         CON(5)  #0E01
           0
1677 E8588 00           CON(2)  0       NO VARIABLES
1678            *
1679            *
1680 E858A 20           CON(2)  2       REGISTER ARITHMETIC
1681 E858C 5            CON(1)  5       OPCODE MNEMONIC
1682 E858D 44D3         NIBASC  \D=C+D \
```



```
           34B2
           4402
1683 E8599 40           CON(2)  (F.TREE)
1684 E859B 3            CON(1)  3       OPCODE
1685 E859C 3000         CON(5)  #003
           0
1686 E85A1 1            CON(1)  1       GROUP C
1687 E85A2 0            CON(1)  0
1688            *
1689            *
1690 E85A3 F0           CON(2)  15      LCHEX
1691 E85A5 5            CON(1)  5       OPCODE MNEMONIC
1692 E85A6 C434         NIBASC  \LCHEX \
           8454
           8502
1693 E85B2 40           CON(2)  (F.TREE)
1694 E85B4 2            CON(1)  2       OPCODE
1695 E85B5 0300         CON(5)  #30
           0
1696 E85BA 3            CON(1)  3       FILL.MARK
1697 E85BB 2            CON(1)  2       FILL.LENGTH
1698            *
1699            *
1700 E85BC 00           CON(2)  0       NULL
1701 E85BE 0            CON(1)  0       OPCODE MNEMONIC
1702 E85BF 0202         NIBASC  \      \
           0202
           0202
1703 E85CB 40           CON(2)  (F.TREE)
1704 E85CD 0            CON(1)  0       OPCODE
1705 E85CE 0000         CON(5)  #0
           0
1706 E85D3 00           CON(2)  0       NO VARIABLES
```

```
1707                    STITLE  HASH TO GROUP 18
1708 E85D5      OPCG18
1709 E85D5 A0           CON(2)  10      SET POINTER
1710 E85D7 3            CON(1)  3       OPCODE MNEMONIC
1711 E85D8 34D3         NIBASC  \C=P   \
           0502
           0202
1712 E85E4 12           CON(2)  (F.LSET)!(F.EXPR)
1713 E85E6 4            CON(1)  4       OPCODE
1714 E85E7 0C08         CON(5)  #80C0
           0
1715 E85EC 00           CON(2)  0       NO VARIABLES
1716            *
1717            *
1718 E85EE 20           CON(2)  2       REGISTER ARITHMETIC
1719 E85F0 4            CON(1)  4       OPCODE MNEMONIC
1720 E85F1 34D3         NIBASC  \C=-C  \
           D234
           0202
1721 E85FD 00           CON(2)  0
1722 E85FF 3            CON(1)  3       OPCODE
1723 E8600 A000         CON(5)  #00A
           0
1724 E8605 4            CON(1)  4       GROUP F
1725 E8606 0            CON(1)  0
1726            *
1727            *
1728 E8607 30           CON(2)  3       REGISTER LOGIC
1729 E8609 5            CON(1)  5       OPCODE MNEMONIC
1730 E860A 44D3         NIBASC  \D=C!D \
           3412
           4402
1731 E8616 50           CON(2)  (F.LSET)!(F.TREE)
1732 E8618 4            CON(1)  4       OPCODE
1733 E8619 B0E0         CON(5)  #0E0B
           0
1734 E861E 00           CON(2)  0       NO VARIABLES
1735            *
1736            *
1737 E8620 50           CON(2)  5       FIXED
1738 E8622 5            CON(1)  5       OPCODE MNEMONIC
1739 E8623 1444         NIBASC  \AD1EX \
           1354
           8502
1740 E862F 50           CON(2)  (F.LSET)!(F.TREE)
1741 E8631 3            CON(1)  3       OPCODE
1742 E8632 3310         CON(5)  #133
           0
1743 E8637 00           CON(2)  0       NO VARIABLES
1744            *
1745            *
1746 E8639 20           CON(2)  2       REGISTER ARITHMETIC
1747 E863B 3            CON(1)  3       OPCODE MNEMONIC
1748 E863C 34D3         NIBASC  \C=0   \
           0302
```



```
           0202
1749 E8648 40           CON(2)  (F.TREE)
1750 E864A 3            CON(1)  3       OPCODE
1751 E864B 2000         CON(5)  #002
           0
1752 E8650 2            CON(1)  2       GROUP D
1753 E8651 0            CON(1)  0
```

```
1754                    STITLE  HASH TO GROUP 19
1755 E8652      OPCG19
1756 E8652 50           CON(2)  5       FIXED
1757 E8654 4            CON(1)  4       OPCODE MNEMONIC
1758 E8655 4413         NIBASC  \D1=C  \
           D334
           0202
1759 E8661 10           CON(2)  (F.LSET)
1760 E8663 3            CON(1)  3       OPCODE
1761 E8664 5310         CON(5)  #135
           0
1762 E8669 00           CON(2)  0       NO VARIABLES
1763            *
1764            *
1765 E866B 20           CON(2)  2       REGISTER ARITHMETIC
1766 E866D 3            CON(1)  3       OPCODE MNEMONIC
1767 E866E 1435         NIBASC  \ASL   \
           C402
           0202
1768 E867A 00           CON(2)  0
1769 E867C 3            CON(1)  3       OPCODE
1770 E867D 0000         CON(5)  #000
           0
1771 E8682 4            CON(1)  4       GROUP F
1772 E8683 0            CON(1)  0
1773            *
1774            *
1775 E8684 40           CON(2)  4       BRANCHES
1776 E8686 6            CON(1)  6       OPCODE MNEMONIC
1777 E8687 74F4         NIBASC  \GOLONG\
           C4F4
           E474
1778 E8693 12           CON(2)  (F.LSET)!(F.EXPR)
1779 E8695 6            CON(1)  6       OPCODE
1780 E8696 000C         CON(5)  #8C000
           8
1781 E869B 3            CON(1)  3       FILL.MARK
1782 E869C 4            CON(1)  4       FILL.LENGTH
1783            *
1784            *
1785 E869D 10           CON(2)  1       REGISTER TESTS
1786 E869F 4            CON(1)  4       OPCODE MNEMONIC
1787 E86A0 F344         NIBASC  \?D<C  \
           C334
           0202
1788 E86AC 70           CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
1789 E86AE 3            CON(1)  3       OPCODE
1790 E86AF 7080         CON(5)  #807
           0
1791 E86B4 2            CON(1)  2       GROUP B
1792 E86B5 0            CON(1)  0
1793            *
1794            *
1795 E86B6 30           CON(2)  3       REGISTER LOGIC
1796 E86B8 5            CON(1)  5       OPCODE MNEMONIC
```



```
1797 E86B9 34D3         NIBASC  \C=C&B \
           3462
           2402
1798 E86C5 50           CON(2)  (F.LSET)!(F.TREE)
1799 E86C7 4            CON(1)  4       OPCODE
1800 E86C8 50E0         CON(5)  #0E05
           0
1801 E86CD 00           CON(2)  0       NO VARIABLES
1802            *
1803            *
1804 E86CF 72           CON(2) 39       PSEUDO-OP
1805 E86D1 6            CON(1) 6        OPCODE MNEMONIC
1806 E86D2 54E4         NIBASC \ENDTXT\
           4445
           8545
1807 E86DE 40           CON(2) (F.TREE)
1808 E86E0 0            CON(1) 0        OPCODE
1809 E86E1 0000         CON(5) #0
           0
1810 E86E6 00           CON(2) 0        NO VARIABLES
1811            *
1812            *
1813 E86E8 00           CON(2) 0        NULL
1814 E86EA 0            CON(1) 0        OPCODE MNEMONIC
1815 E86EB 0202         NIBASC \      \
           0202
           0202
1816 E86F7 40           CON(2) (F.TREE)
1817 E86F9 0            CON(1) 0        OPCODE
1818 E86FA 0000         CON(5) #0
           0
1819 E86FF 00           CON(2) 0        NO VARIABLES
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER

```
1820                    STITLE  HASH TO GROUP 20
1821 E8701      OPCG20
1822 E8701 C1           CON(2)  28      PSEUDO-OP
1823 E8703 3            CON(1)  3       OPCODE MNEMONIC
1824 E8704 C454         NIBASC  \LEX   \
           8502
           0202
1825 E8710 02           CON(2)  (F.EXPR)
1826 E8712 0            CON(1)  0       OPCODE
1827 E8713 0000         CON(5)  #0
           0
1828 E8718 00           CON(2)  0       NO VARIABLES
1829            *
1830            *
1831 E871A 50           CON(2)  5       FIXED
1832 E871C 4            CON(1)  4       OPCODE MNEMONIC
1833 E871D 4413         NIBASC  \D1=A  \
           D314
           0202
1834 E8729 10           CON(2)  (F.LSET)
1835 E872B 3            CON(1)  3       OPCODE
1836 E872C 1310         CON(5)  #131
           0
1837 E8731 00           CON(2)  0       NO VARIABLES
1838            *
1839            *
1840 E8733 11           CON(2)  17      NIBASC STATEMENT
1841 E8735 6            CON(1)  6       OPCODE MNEMONIC
1842 E8736 E494         NIBASC  \NIBASC\
           2414
           3534
1843 E8742 42           CON(2)  (F.TREE)!(F.EXPR)
1844 E8744 0            CON(1)  0       OPCODE
1845 E8745 0000         CON(5)  #0
           0
1846 E874A 1            CON(1)  1       FILL.MARK
1847 E874B E            CON(1)  14      FILL.LENGTH
1848            *
1849            *
1850 E874C 10           CON(2)  1       REGISTER TESTS
1851 E874E 5            CON(1)  5       OPCODE MNEMONIC
1852 E874F F324         NIBASC  \?B<=C \
           C3D3
           3402
1853 E875B 70           CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
1854 E875D 3            CON(1)  3       OPCODE
1855 E875E D080         CON(5)  #80D
           0
1856 E8763 2            CON(1)  2       GROUP B
1857 E8764 0            CON(1)  0
1858            *
1859            *
1860 E8765 20           CON(2)  2       REGISTER ARITHMETIC
1861 E8767 3            CON(1)  3       OPCODE MNEMONIC
1862 E8768 44D3         NIBASC  \D=0   \
```



```
           0302
           0202
1863 E8774 40           CON(2)  (F.TREE)
1864 E8776 3            CON(1)  3       OPCODE
1865 E8777 3000         CON(5)  #003
           0
1866 E877C 2            CON(1)  2       GROUP D
1867 E877D 0            CON(1)  0
```

```
1868                    STITLE  HASH TO GROUP 21
1869 E877E      OPCG21
1870 E877E 40           CON(2)  4       BRANCHES
1871 E8780 6            CON(1)  6       OPCODE MNEMONIC
1872 E8781 4403         NIBASC  \D0=(4)\
           D382
           4392
1873 E878D 1B           CON(2)  (F.LSET)!(F.ABSL)!(F.EXPR)!(F.PART)
1874 E878F 6            CON(1)  6       OPCODE
1875 E8790 000A         CON(5)  #1A000
           1
1876 E8795 3            CON(1)  3       FILL.MARK
1877 E8796 4            CON(1)  4       FILL.LENGTH
1878            *
1879            *
1880 E8797 10           CON(2)  1       REGISTER TESTS
1881 E8799 4            CON(1)  4       OPCODE MNEMONIC
1882 E879A F334         NIBASC  \?C>D  \
           E344
           0202
1883 E87A6 30           CON(2)  (F.LSET)!(F.RETY)
1884 E87A8 3            CON(1)  3       OPCODE
1885 E87A9 7080         CON(5)  #807
           0
1886 E87AE 2            CON(1)  2       GROUP B
1887 E87AF 0            CON(1)  0
1888            *
1889            *
1890 E87B0 41           CON(2)  20      EJECT PSEUDO-OP
1891 E87B2 5            CON(1)  5       OPCODE MNEMONIC
1892 E87B3 54A4         NIBASC  \EJECT \
           5434
           4502
1893 E87BF 10           CON(2)  (F.LSET)
1894 E87C1 0            CON(1)  0       OPCODE
1895 E87C2 0000         CON(5)  #0
           0
1896 E87C7 00           CON(2)  0       NO VARIABLES
1897            *
1898            *
1899 E87C9 10           CON(2)  1       REGISTER TESTS
1900 E87CB 4            CON(1)  4       OPCODE MNEMONIC
1901 E87CC F314         NIBASC  \?A>C  \
           E334
           0202
1902 E87D8 70           CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
1903 E87DA 3            CON(1)  3       OPCODE
1904 E87DB 6080         CON(5)  #806
           0
1905 E87E0 2            CON(1)  2       GROUP B
1906 E87E1 0            CON(1)  0
1907            *
1908            *
1909 E87E2 20           CON(2)  2       REGISTER ARITHMETIC
1910 E87E4 3            CON(1)  3       OPCODE MNEMONIC
```



```
1911 E87E5 2435         NIBASC  \BSL   \
           C402
           0202
1912 E87F1 40           CON(2)  (F.TREE)
1913 E87F3 3            CON(1)  3       OPCODE
1914 E87F4 1000         CON(5)  #001
           0
1915 E87F9 4            CON(1)  4       GROUP F
1916 E87FA 0            CON(1)  0
1917            *
1918            *
1919 E87FB 20           CON(2)  2       REGISTER ARITHMETIC
1920 E87FD 4            CON(1)  4       OPCODE MNEMONIC
1921 E87FE 4434         NIBASC  \DCEX  \
           5485
           0202
1922 E880A 40           CON(2)  (F.TREE)
1923 E880C 3            CON(1)  3       OPCODE
1924 E880D F000         CON(5)  #00F
           0
1925 E8812 2            CON(1)  2       GROUP D
1926 E8813 0            CON(1)  0
1927            *
1928            *
1929 E8814 00           CON(2)  0       NULL
1930 E8816 0            CON(1)  0       OPCODE MNEMONIC
1931 E8817 0202         NIBASC  \      \
           0202
           0202
1932 E8823 40           CON(2)  (F.TREE)
1933 E8825 0            CON(1)  0       OPCODE
1934 E8826 0000         CON(5)  #0
           0
1935 E882B 00           CON(2)  0       NO VARIABLES
```

```
1936                    STITLE  HASH TO GROUP 22
1937 E882D      OPCG22
1938 E882D 50           CON(2)  5       FIXED
1939 E882F 5            CON(1)  5       OPCODE MNEMONIC
1940 E8830 3444         NIBASC  \CD1EX \
           1354
           8502
1941 E883C 10           CON(2)  (F.LSET)
1942 E883E 3            CON(1)  3       OPCODE
1943 E883F 7310         CON(5)  #137
           0
1944 E8844 00           CON(2)  0       NO VARIABLES
1945            *
1946            *
1947 E8846 50           CON(2)  5       FIXED
1948 E8848 5            CON(1)  5       OPCODE MNEMONIC
1949 E8849 1444         NIBASC  \AD0EX \
           0354
           8502
1950 E8855 10           CON(2)  (F.LSET)
1951 E8857 3            CON(1)  3       OPCODE
1952 E8858 2310         CON(5)  #132
           0
1953 E885D 00           CON(2)  0       NO VARIABLES
1954            *
1955            *
1956 E885F 50           CON(2)  5       FIXED
1957 E8861 4            CON(1)  4       OPCODE MNEMONIC
1958 E8862 E4F4         NIBASC  \NOP5  \
           0553
           0202
1959 E886E 10           CON(2)  (F.LSET)
1960 E8870 5            CON(1)  5       OPCODE
1961 E8871 0004         CON(5)  #64000
           6
1962 E8876 00           CON(2)  0       NO VARIABLES
1963            *
1964            *
1965 E8878 10           CON(2)  1       REGISTER TESTS
1966 E887A 4            CON(1)  4       OPCODE MNEMONIC
1967 E887B F334         NIBASC  \?C>B  \
           E324
           0202
1968 E8887 70           CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
1969 E8889 3            CON(1)  3       OPCODE
1970 E888A 5080         CON(5)  #805
           0
1971 E888F 2            CON(1)  2       GROUP B
1972 E8890 0            CON(1)  0
1973            *
1974            *
1975 E8891 50           CON(2)  5        FIXED
1976 E8893 6            CON(1)  6       OPCODE MNEMONIC
1977 E8894 34D3         NIBASC  \C=RSTK\
           2535
```



```
           45B4
1978 E88A0 50           CON(2)  (F.LSET)!(F.TREE)
1979 E88A2 2            CON(1)  2       OPCODE
1980 E88A3 7000         CON(5)  #07
           0
1981 E88A8 00           CON(2)  0       NO VARIABLES
1982            *
1983            *
1984 E88AA D0           CON(2)  13      DATA TRANSFER
1985 E88AC 6            CON(1)  6       OPCODE MNEMONIC
1986 E88AD 4414         NIBASC  \DAT1=A\
           4513
           D314
1987 E88B9 42           CON(2)  (F.TREE)!(F.EXPR)
1988 E88BB 4            CON(1)  4       OPCODE
1989 E88BC 0151         CON(5)  #1510
           0
1990 E88C1 00           CON(2)  0       NO VARIABLES
1991            *
1992            *
1993 E88C3 50           CON(2)  5       FIXED
1994 E88C5 6            CON(1)  6       OPCODE MNEMONIC
1995 E88C6 3584         NIBASC  \SHUTDN\
           5545
           44E4
1996 E88D2 50           CON(2)  (F.LSET)!(F.TREE)
1997 E88D4 3            CON(1)  3       OPCODE
1998 E88D5 7080         CON(5)  #807
           0
1999 E88DA 00           CON(2)  0       NO VARIABLES
```

```
2000                    STITLE  HASH TO GROUP 23
2001 E88DC      OPCG23
2002 E88DC 50           CON(2)  5       FIXED
2003 E88DE 5            CON(1)  5       OPCODE MNEMONIC
2004 E88DF 1444         NIBASC  \AD1XS \
           1385
           3502
2005 E88EB 10           CON(2)  (F.LSET)
2006 E88ED 3            CON(1)  3       OPCODE
2007 E88EE B310         CON(5)  #13B
           0
2008 E88F3 00           CON(2)  0       NO VARIABLES
2009            *
2010            *
2011 E88F5 10           CON(2)  1       REGISTER TESTS
2012 E88F7 4            CON(1)  4       OPCODE MNEMONIC
2013 E88F8 F344         NIBASC  \?D#C  \
           3234
           0202
2014 E8904 30           CON(2)  (F.LSET)!(F.RETY)
2015 E8906 3            CON(1)  3       OPCODE
2016 E8907 7080         CON(5)  #807
           0
2017 E890C 1            CON(1)  1       GROUP A
2018 E890D 0            CON(1)  0
2019            *
2020            *
2021 E890E 20           CON(2)  2       REGISTER ARITHMETIC
2022 E8910 3            CON(1)  3       OPCODE MNEMONIC
2023 E8911 3435         NIBASC  \CSL   \
           C402
           0202
2024 E891D 00           CON(2)  0
2025 E891F 3            CON(1)  3       OPCODE
2026 E8920 2000         CON(5)  #002
           0
2027 E8925 4            CON(1)  4       GROUP F
2028 E8926 0            CON(1)  0
2029            *
2030            *
2031 E8927 10           CON(2)  1       REGISTER TESTS
2032 E8929 5            CON(1)  5       OPCODE MNEMONIC
2033 E892A F334         NIBASC  \?C<=B \
           C3D3
           2402
2034 E8936 70           CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
2035 E8938 3            CON(1)  3       OPCODE
2036 E8939 9080         CON(5)  #809
           0
2037 E893E 2            CON(1)  2       GROUP B
2038 E893F 0            CON(1)  0
2039            *
2040            *
2041 E8940 30           CON(2)  3       REGISTER LOGIC
2042 E8942 5            CON(1)  5       OPCODE MNEMONIC
```



```
2043 E8943 14D3         NIBASC  \A=A&B \
           1462
           2402
2044 E894F 50           CON(2)  (F.LSET)!(F.TREE)
2045 E8951 4            CON(1)  4       OPCODE
2046 E8952 00E0         CON(5)  #0E00
           0
2047 E8957 00           CON(2)  0       NO VARIABLES
2048            *
2049            *
2050 E8959 30           CON(2)  3       REGISTER LOGIC
2051 E895B 5            CON(1)  5       OPCODE MNEMONIC
2052 E895C 34D3         NIBASC  \C=D&C \
           4462
           3402
2053 E8968 50           CON(2)  (F.LSET)!(F.TREE)
2054 E896A 4            CON(1)  4       OPCODE
2055 E896B 70E0         CON(5)  #0E07
           0
2056 E8970 00           CON(2)  0       NO VARIABLES
2057            *
2058            *
2059 E8972 50           CON(2)  5       FIXED
2060 E8974 4            CON(1)  4       OPCODE MNEMONIC
2061 E8975 E4F4         NIBASC  \NOP3  \
           0533
           0202
2062 E8981 50           CON(2)  (F.LSET)!(F.TREE)
2063 E8983 3            CON(1)  3       OPCODE
2064 E8984 0240         CON(5)  #420
           0
2065 E8989 00           CON(2)  0       NO VARIABLES
```

```
2066                    STITLE  HASH TO GROUP 24
2067 E898B      OPCG24
2068 E898B 02           CON(2)  32      PSEUDO-OP
2069 E898D 5            CON(1)  5       OPCODE MNEMONIC
2070 E898E 54E4         NIBASC  \ENTRY \
           4525
           9502
2071 E899A 12           CON(2)  (F.LSET)!(F.EXPR)
2072 E899C 8            CON(1)  8       OPCODE
2073 E899D 0000         CON(5)  #0
           0
2074 E89A2 00           CON(2)  0       NO VARIABLES
2075            *
2076            *
2077 E89A4 50           CON(2)  5       FIXED
2078 E89A6 5            CON(1)  5       OPCODE MNEMONIC
2079 E89A7 34B2         NIBASC  \C+P+1 \
           05B2
           1302
2080 E89B3 10           CON(2)  (F.LSET)
2081 E89B5 3            CON(1)  3       OPCODE
2082 E89B6 9080         CON(5)  #809
           0
2083 E89BB 00           CON(2)  0       NO VARIABLES
2084            *
2085            *
2086 E89BD 50           CON(2)  5       FIXED
2087 E89BF 3            CON(1)  3       OPCODE MNEMONIC
2088 E89C0 2545         NIBASC  \RTN   \
           E402
           0202
2089 E89CC 50           CON(2)  (F.LSET)!(F.TREE)
2090 E89CE 2            CON(1)  2       OPCODE
2091 E89CF 1000         CON(5)  #01
           0
2092 E89D4 00           CON(2)  0       NO VARIABLES
2093            *
2094            *
2095 E89D6 10           CON(2)  1       REGISTER TESTS
2096 E89D8 4            CON(1)  4       OPCODE MNEMONIC
2097 E89D9 F324         NIBASC  \?B#0  \
           3203
           0202
2098 E89E5 70           CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
2099 E89E7 3            CON(1)  3       OPCODE
2100 E89E8 D080         CON(5)  #80D
           0
2101 E89ED 1            CON(1)  1       GROUP A
2102 E89EE 0            CON(1)  0
2103            *
2104            *
2105 E89EF D0           CON(2)  13      DATA TRANSFER
2106 E89F1 6            CON(1)  6       OPCODE MNEMONIC
2107 E89F2 4414         NIBASC  \DAT1=C\
           4513
```



```
           D334
2108 E89FE 42           CON(2)  (F.TREE)!(F.EXPR)
2109 E8A00 4            CON(1)  4       OPCODE
2110 E8A01 0551         CON(5)  #1550
           0
2111 E8A06 00           CON(2)  0       NO VARIABLES
```

```
2112                    STITLE  HASH TO GROUP 25
2113 E8A08      OPCG25
2114 E8A08 40           CON(2)  4       BRANCHES
2115 E8A0A 5            CON(1)  5       OPCODE MNEMONIC
2116 E8A0B C434         NIBASC  \LC(5) \
          8253
          9202
2117 E8A17 1B           CON(2)  (F.LSET)!(F.ABSL)!(F.EXPR)!(F.PART)
2118 E8A19 7            CON(1)  7       OPCODE
2119 E8A1A 0004         CON(5)  #34000
          3
2120 E8A1F 3            CON(1)  3       FILL MARK
2121 E8A20 5            CON(1)  5       FILL LENGTH
2122            *
2123            *
2124 E8A21 31           CON(2)  19      BSS PSEUDO-OP
2125 E8A23 3            CON(1)  3       OPCODE MNEMONIC
2126 E8A24 2435         NIBASC  \BSS   \
          3502
          0202
2127 E8A30 02           CON(2)  (F.EXPR)
2128 E8A32 0            CON(1)  0       OPCODE
2129 E8A33 0000         CON(5)  #0
          0
2130 E8A38 00           CON(2)  0       NO VARIABLES
2131            *
2132            *
2133 E8A3A 50           CON(2)  5       FIXED
2134 E8A3C 5            CON(1)  5       OPCODE MNEMONIC
2135 E8A3D 2545         NIBASC  \RTNCC \
          E434
          3402
2136 E8A49 10           CON(2)  (F.LSET)
2137 E8A4B 2            CON(1)  2       OPCODE
2138 E8A4C 3000         CON(5)  #03
          0
2139 E8A51 00           CON(2)  0       NO VARIABLES
2140            *
2141            *
2142 E8A53 10           CON(2)  1       REGISTER TESTS
2143 E8A55 4            CON(1)  4       OPCODE MNEMONIC
2144 E8A56 F334         NIBASC  \?C=D  \
          D344
          0202
2145 E8A62 30           CON(2)  (F.LSET)!(F.RETY)
2146 E8A64 3            CON(1)  3       OPCODE
2147 E8A65 3080         CON(5)  #803
          0
2148 E8A6A 1            CON(1)  1       GROUP A
2149 E8A6B 0            CON(1)  0
2150            *
2151            *
2152 E8A6C 30           CON(2)  3       REGISTER LOGIC
2153 E8A6E 5            CON(1)  5       OPCODE MNEMONIC
2154 E8A6F 44D3         NIBASC  \D=D&C \
```



```
          4462
          3402
2155 E8A7B 10           CON(2)  (F.LSET)
2156 E8A7D 4            CON(1)  4       OPCODE
2157 E8A7E 30E0         CON(5)  #0E03
          0
2158 E8A83 00           CON(2)  0       NO VARIABLES
2159            *
2160            *
2161 E8A85 40           CON(2)  4       BRANCHES
2162 E8A87 6            CON(1)  6       OPCODE MNEMONIC
2163 E8A88 2554         NIBASC  \REL(3)\
          C482
          3392
2164 E8A94 52           CON(2)  (F.LSET)!(F.TREE)!(F.EXPR)
2165 E8A96 3            CON(1)  3       OPCODE
2166 E8A97 0000         CON(5)  #0
          0
2167 E8A9C 1            CON(1)  1       FILL.MARK
2168 E8A9D 3            CON(1)  3       FILL.LENGTH
2169            *
2170            *
2171 E8A9E 50           CON(2)  5       FIXED
2172 E8AA0 5            CON(1)  5       OPCODE MNEMONIC
2173 E8AA1 2545         NIBASC  \RTNSC \
          E435
          3402
2174 E8AAD 50           CON(2)  (F.LSET)!(F.TREE)
2175 E8AAF 2            CON(1)  2       OPCODE
2176 E8AB0 2000         CON(5)  #02
          0
2177 E8AB5 00           CON(2)  0       NO VARIABLES
2178            *
2179            *
2180 E8AB7 10           CON(2)  1       REGISTER TESTS
2181 E8AB9 4            CON(1)  4       OPCODE MNEMONIC
2182 E8ABA F314         NIBASC  \?A=C  \
          D334
          0202
2183 E8AC6 70           CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
2184 E8AC8 3            CON(1)  3       OPCODE
2185 E8AC9 2080         CON(5)  #802
          0
2186 E8ACE 1            CON(1)  1       GROUP A
2187 E8ACF 0            CON(1)  0
2188            *
2189            *
2190 E8AD0 30           CON(2)  3       REGISTER LOGIC
2191 E8AD2 5            CON(1)  5       OPCODE MNEMONIC
2192 E8AD3 24D3         NIBASC  \B=A&B \
          1462
          2402
2193 E8ADF 50           CON(2)  (F.LSET)!(F.TREE)
2194 E8AE1 4            CON(1)  4       OPCODE
2195 E8AE2 40E0         CON(5)  #0E04
```

```
          0
2196 E8AE7 00          CON(2)  0       NO VARIABLES
2197             *
2198             *
2199 E8AE9 40          CON(2)  4       BRANCHES
2200 E8AEB 6           CON(1)  6       OPCODE MNEMONIC
2201 E8AEC 34F4        NIBASC  \CON(1)\
           E482
           1392
2202 E8AF8 53          CON(2)  (F.LSET)!(F.TREE)!(F.ABSL)!(F.EXPR)
2203 E8AFA 1           CON(1)  1       OPCODE
2204 E8AFB 0000        CON(5)  #0
           0
2205 E8B00 1           CON(1)  1       FILL.MARK
2206 E8B01 1           CON(1)  1       FILL.LENGTH
2207             *
2208             *
2209 E8B02 20          CON(2)  2       REGISTER ARITHMETIC
2210 E8B04 3           CON(1)  3       OPCODE MNEMONIC
2211 E8B05 4435        NIBASC  \DSL    \
           C402
           0202
2212 E8B11 40          CON(2)  (F.TREE)
2213 E8B13 3           CON(1)  3       OPCODE
2214 E8B14 3000        CON(5)  #003
           0
2215 E8B19 4           CON(1)  4       GROUP F
2216 E8B1A 0           CON(1)  0
```



```
2217                   STITLE  HASH TO GROUP 26
2218 E8B1B       OPCG26
2219 E8B1B 50          CON(2)  5       FIXED
2220 E8B1D 5           CON(1)  5       OPCODE MNEMONIC
2221 E8B1E 3444        NIBASC  \CD0EX \
           0354
           8502
2222 E8B2A 10          CON(2)  (F.LSET)
2223 E8B2C 3           CON(1)  3       OPCODE
2224 E8B2D 6310        CON(5)  #136
           0
2225 E8B32 00          CON(2)  0       NO VARIABLES
2226             *
2227             *
2228 E8B34 50          CON(2)  5       FIXED
2229 E8B36 4           CON(1)  4       OPCODE MNEMONIC
2230 E8B37 14D3        NIBASC  \A=R3  \
           2533
           0202
2231 E8B43 10          CON(2)  (F.LSET)
2232 E8B45 3           CON(1)  3       OPCODE
2233 E8B46 3110        CON(5)  #113
           0
2234 E8B4B 00          CON(2)  0       NO VARIABLES
2235             *
2236             *
2237 E8B4D 50          CON(2)  5       FIXED
2238 E8B4F 4           CON(1)  4       OPCODE MNEMONIC
2239 E8B50 2523        NIBASC  \R2=C  \
           D334
           0202
2240 E8B5C 10          CON(2)  (F.LSET)
2241 E8B5E 3           CON(1)  3       OPCODE
2242 E8B5F A010        CON(5)  #10A
           0
2243 E8B64 00          CON(2)  0       NO VARIABLES
2244             *
2245             *
2246 E8B66 10          CON(2)  1       REGISTER TESTS
2247 E8B68 4           CON(1)  4       OPCODE MNEMONIC
2248 E8B69 F334        NIBASC  \?C=B  \
           D324
           0202
2249 E8B75 70          CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
2250 E8B77 3           CON(1)  3       OPCODE
2251 E8B78 1080        CON(5)  #801
           0
2252 E8B7D 1           CON(1)  1       GROUP A
2253 E8B7E 0           CON(1)  0
2254             *
2255             *
2256 E8B7F 50          CON(2)  5       FIXED
2257 E8B81 4           CON(1)  4       OPCODE MNEMONIC
2258 E8B82 1435        NIBASC  \ASRB  \
           2524
```

```
            0202
2259 E8B8E 50          CON(2)  (F.LSET)!(F.TREE)
2260 E8B90 3           CON(1)  3       OPCODE
2261 E8B91 C180        CON(5)  #81C
           0
2262 E8B96 00          CON(2)  0       NO VARIABLES
2263            *
2264            *
2265 E8B98 40          CON(2)  4       BRANCHES
2266 E8B9A 5           CON(1)  5       OPCODE MNEMONIC
2267 E8B9B C434        NIBASC  \LC(3) \
           8233
           9202
2268 E8BA7 5B          CON(2)  (F.LSET)!(F.ABSL)!(F.TREE)!(F.EXPR)!(F.PART)
2269 E8BA9 5           CON(1)  5       OPCODE
2270 E8BAA 0002        CON(5)  #32000
           3
2271 E8BAF 3           CON(1)  3       FILL.MARK
2272 E8BB0 3           CON(1)  3       FILL.LENGTH
2273            *
2274            *
2275 E8BB1 00          CON(2)  0       NULL
2276 E8BB3 0           CON(1)  0       OPCODE MNEMONIC
2277 E8BB4 0202        NIBASC  \      \
           0202
           0202
2278 E8BC0 40          CON(2)  (F.TREE)
2279 E8BC2 0           CON(1)  0       OPCODE
2280 E8BC3 0000        CON(5)  #0
           0
2281 E8BC8 00          CON(2)  0       NO VARIABLES
```



```
2282                    STITLE  HASH TO GROUP 27
2283 E8BCA      OPCG27
2284 E8BCA 50          CON(2)  5       FIXED
2285 E8BCC 5           CON(1)  5       OPCODE MNEMONIC
2286 E8BCD 3444        NIBASC  \CD1XS \
           1385
           3502
2287 E8BD9 10          CON(2)  (F.LSET)
2288 E8BDB 3           CON(1)  3       OPCODE
2289 E8BDC F310        CON(5)  #13F
           0
2290 E8BE1 00          CON(2)  0       NO VARIABLES
2291            *
2292            *
2293 E8BE3 50          CON(2)  5       FIXED
2294 E8BE5 5           CON(1)  5       OPCODE MNEMONIC
2295 E8BE6 1425        NIBASC  \AR4EX \
           4354
           8502
2296 E8BF2 10          CON(2)  (F.LSET)
2297 E8BF4 3           CON(1)  3       OPCODE
2298 E8BF5 4210        CON(5)  #124
           0
2299 E8BFA 00          CON(2)  0       NO VARIABLES
2300            *
2301            *
2302 E8BFC 40          CON(2)  4       BRANCHES
2303 E8BFE 5           CON(1)  5       OPCODE MNEMONIC
2304 E8BFF C434        NIBASC  \LC(1) \
           8213
           9202
2305 E8C0B 1B          CON(2)  (F.LSET)!(F.ABSL)!(F.EXPR)!(F.PART)
2306 E8C0D 3           CON(1)  3       OPCODE
2307 E8C0E 0030        CON(5)  #300
           0
2308 E8C13 3           CON(1)  3       FILL.MARK
2309 E8C14 1           CON(1)  1       FILL.LENGTH
2310            *
2311            *
2312 E8C15 50          CON(2)  5       FIXED
2313 E8C17 4           CON(1)  4       OPCODE MNEMONIC
2314 E8C18 14D3        NIBASC  \A=R1  \
           2513
           0202
2315 E8C24 10          CON(2)  (F.LSET)
2316 E8C26 3           CON(1)  3       OPCODE
2317 E8C27 1110        CON(5)  #111
           0
2318 E8C2C 00          CON(2)  0       NO VARIABLES
2319            *
2320            *
2321 E8C2E 20          CON(2)  2       REGISTER ARITHMETIC
2322 E8C30 3           CON(1)  3       OPCODE MNEMONIC
2323 E8C31 1435        NIBASC  \ASR   \
           2502
```

```
            0202
2324 E8C3D 40          CON(2)  (F.TREE)
2325 E8C3F 3           CON(1)  3       OPCODE
2326 E8C40 4000        CON(5)  #004
           0
2327 E8C45 4           CON(1)  4       GROUP F
2328 E8C46 0           CON(1)  0
2329           *
2330           *
2331 E8C47 50          CON(2)  5       FIXED
2332 E8C49 6           CON(1)  6       OPCODE MNEMONIC
2333 E8C4A 34C4        NIBASC  \CLRHST\
           2584
           3545
2334 E8C56 50          CON(2)  (F.LSET)!(F.TREE)
2335 E8C58 3           CON(1)  3       OPCODE
2336 E8C59 F280        CON(5)  #82F
           0
2337 E8C5E 00          CON(2)  0       NO VARIABLES
2338           *
2339           *
2340 E8C60 50          CON(2)  5       FIXED
2341 E8C62 4           CON(1)  4       OPCODE MNEMONIC
2342 E8C63 2523        NIBASC  \R2=A  \
           D314
           0202
2343 E8C6F 50          CON(2)  (F.LSET)!(F.TREE)
2344 E8C71 3           CON(1)  3       OPCODE
2345 E8C72 2010        CON(5)  #102
           0
2346 E8C77 00          CON(2)  0       NO VARIABLES
2347           *
2348           *
2349 E8C79 10          CON(2)  1       REGISTER TESTS
2350 E8C7B 4           CON(1)  4       OPCODE MNEMONIC
2351 E8C7C F334        NIBASC  \?C=0  \
           D303
           0202
2352 E8C88 70          CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
2353 E8C8A 3           CON(1)  3       OPCODE
2354 E8C8B A080        CON(5)  #80A
           0
2355 E8C90 1           CON(1)  1       GROUP A
2356 E8C91 0           CON(1)  0
2357           *
2358           *
2359 E8C92 50          CON(2)  5       FIXED
2360 E8C94 5           CON(1)  5       OPCODE MNEMONIC
2361 E8C95 1444        NIBASC  \ADOXS \
           0385
           3502
2362 E8CA1 50          CON(2)  (F.LSET)!(F.TREE)
2363 E8CA3 3           CON(1)  3       OPCODE
2364 E8CA4 A310        CON(5)  #13A
           0
```



```
2365 E8CA9 00          CON(2)  0       NO VARIABLES
```

```
2366                    STITLE  HASH TO GROUP 28
2367 E8CAB      OPCG28
2368 E8CAB 01           CON(2)  16      Dx=HEX INSTRUCTIONS
2369 E8CAD 6            CON(1)  6       OPCODE MNEMONIC
2370 E8CAE 4403         NIBASC  \D0=HEX\
          D384
          5485
2371 E8CBA 10           CON(2)  (F.LSET)
2372 E8CBC 4            CON(1)  4       OPCODE
2373 E8CBD 0091         CON(5)  #1900
          0
2374 E8CC2 3            CON(1)  3       FILL.MARK
2375 E8CC3 2            CON(1)  2       FILL.LENGTH
2376            *
2377            *
2378 E8CC4 62           CON(2)  38      CHAIN
2379 E8CC6 5            CON(1)  5       OPCODE MNEMONIC
2380 E8CC7 3484         NIBASC  \CHAIN \
          1494
          E402
2381 E8CD3 12           CON(2)  (F.LSET)!(F.EXPR)
2382 E8CD5 C            CON(1)  12      OPCODE
2383 E8CD6 0000         CON(5)  #0
          0
2384 E8CDB 00           CON(2)  0       NO VARIABLES
2385            *
2386            *
2387 E8CDD 40           CON(2)  4       BRANCHES
2388 E8CDF 5            CON(1)  5       OPCODE MNEMONIC
2389 E8CE0 74F4         NIBASC  \GOSUB \
          3555
          2402
2390 E8CEC 56           CON(2)  (F.LSET)!(F.TREE)!(F.EXPR)!(F.GSUB)
2391 E8CEE 4            CON(1)  4       OPCODE
2392 E8CEF 0007         CON(5)  #7000
          0
2393 E8CF4 2            CON(1)  2       FILL.MARK
2394 E8CF5 3            CON(1)  3       FILL.LENGTH
2395            *
2396            *
2397 E8CF6 50           CON(2)  5       FIXED
2398 E8CF8 4            CON(1)  4       OPCODE MNEMONIC
2399 E8CF9 2435         NIBASC  \BSRB  \
          2524
          0202
2400 E8D05 50           CON(2)  (F.LSET)!(F.TREE)
2401 E8D07 3            CON(1)  3       OPCODE
2402 E8D08 D180         CON(5)  #81D
          0
2403 E8D0D 00           CON(2)  0       NO VARIABLES
2404            *
2405            *
2406 E8D0F 50           CON(2)  5       FIXED
2407 E8D11 6            CON(1)  6       OPCODE MNEMONIC
2408 E8D12 34F4         NIBASC  \CONFIG\
```



```
          E464
          9474
2409 E8D1E 50           CON(2)  (F.LSET)!(F.TREE)
2410 E8D20 3            CON(1)  3       OPCODE
2411 E8D21 5080         CON(5)  #805
          0
2412 E8D26 00           CON(2)  0       NO VARIABLES
```

```
2413                    STITLE  HASH TO GROUP 29
2414 E8D28      OPCG29
2415 E8D28 20           CON(2)  2       REGISTER ARITHMETIC
2416 E8D2A 3            CON(1)  3       OPCODE MNEMONIC
2417 E8D2B 2435         NIBASC  \BSR   \
           2502
           0202
2418 E8D37 00           CON(2)  0
2419 E8D39 3            CON(1)  3       OPCODE
2420 E8D3A 5000         CON(5)  #005
           0
2421 E8D3F 4            CON(1)  4       GROUP F
2422 E8D40 0            CON(1)  0
2423            *
2424            *
2425 E8D41 10           CON(2)  1       REGISTER TESTS
2426 E8D43 4            CON(1)  4       OPCODE MNEMONIC
2427 E8D44 F334         NIBASC  \?C<D  \
           C344
           0202
2428 E8D50 30           CON(2)  (F.LSET)!(F.RETY)
2429 E8D52 3            CON(1)  3       OPCODE
2430 E8D53 3080         CON(5)  #803
           0
2431 E8D58 2            CON(1)  2       GROUP B
2432 E8D59 0            CON(1)  0
2433            *
2434            *
2435 E8D5A 40           CON(2)  4       BRANCHES
2436 E8D5C 6            CON(1)  6       OPCODE MNEMONIC
2437 E8D5D 4403         NIBASC  \D0=(5)\
           D382
           5392
2438 E8D69 1B           CON(2)  (F.LSET)!(F.ABSL)!(F.EXPR)!(F.PART)
2439 E8D6B 7            CON(1)  7       OPCODE
2440 E8D6C 000B         CON(5)  #1B000
           1
2441 E8D71 3            CON(1)  3       FILL.MARK
2442 E8D72 5            CON(1)  5       FILL.LENGTH
2443            *
2444            *
2445 E8D73 10           CON(2)  1       REGISTER TESTS
2446 E8D75 4            CON(1)  4       OPCODE MNEMONIC
2447 E8D76 F314         NIBASC  \?A<C  \
           C334
           0202
2448 E8D82 70           CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
2449 E8D84 3            CON(1)  3       OPCODE
2450 E8D85 2080         CON(5)  #802
           0
2451 E8D8A 2            CON(1)  2       GROUP B
2452 E8D8B 0            CON(1)  0
2453            *
2454            *
2455 E8D8C 30           CON(2)  3       REGISTER LOGIC
```



```
2456 E8D8E 5            CON(1)  5       OPCODE MNEMONIC
2457 E8D8F 24D3         NIBASC  \B=B&C \
           2462
           3402
2458 E8D9B 50           CON(2)  (F.LSET)!(F.TREE)
2459 E8D9D 4            CON(1)  4       OPCODE
2460 E8D9E 10E0         CON(5)  #0E01
           0
2461 E8DA3 00           CON(2)  0       NO VARIABLES
2462            *
2463            *
2464 E8DA5 A0           CON(2)  10      SET POINTER
2465 E8DA7 4            CON(1)  4       OPCODE MNEMONIC
2466 E8DA8 3405         NIBASC  \CPEX  \
           5485
           0202
2467 E8DB4 52           CON(2)  (F.LSET)!(F.TREE)!(F.EXPR)
2468 E8DB6 4            CON(1)  4       OPCODE
2469 E8DB7 0F08         CON(5)  #80F0
           0
2470 E8DBC 00           CON(2)  0       NO VARIABLES
2471            *
2472            *
2473 E8DBE 00           CON(2)  0       NULL
2474 E8DC0 0            CON(1)  0       OPCODE MNEMONIC
2475 E8DC1 0202         NIBASC  \      \
           0202
           0202
2476 E8DCD 40           CON(2)  (F.TREE)
2477 E8DCF 0            CON(1)  0       OPCODE
2478 E8DD0 0000         CON(5)  #0
           0
2479 E8DD5 00           CON(2)  0       NO VARIABLES
```

```
2480                    STITLE  HASH TO GROUP 30
2481 E8DD7      OPCG30
2482 E8DD7 20           CON(2)  2       REGISTER ARITHMETIC
2483 E8DD9 4            CON(1)  4       OPCODE MNEMONIC
2484 E8DDA 3444         NIBASC  \CDEX  \
           5485
           0202
2485 E8DE6 00           CON(2)  0
2486 E8DE8 3            CON(1)  3       OPCODE
2487 E8DE9 F000         CON(5)  #00F
           0
2488 E8DEE 2            CON(1)  2       GROUP D
2489 E8DEF 0            CON(1)  0
2490            *
2491            *
2492 E8DF0 20           CON(2)  2       REGISTER ARITHMETIC
2493 E8DF2 5            CON(1)  5       OPCODE MNEMONIC
2494 E8DF3 14D3         NIBASC  \A=C+A \
           34B2
           1402
2495 E8DFF 00           CON(2)  0
2496 E8E01 3            CON(1)  3       OPCODE
2497 E8E02 A000         CON(5)  #00A
           0
2498 E8E07 1            CON(1)  1       GROUP C
2499 E8E08 0            CON(1)  0
2500            *
2501            *
2502 E8E09 B0           CON(2)  11      SET STATUS
2503 E8E0B 4            CON(1)  4       OPCODE MNEMONIC
2504 E8E0C 3545         NIBASC  \ST=0  \
           D303
           0202
2505 E8E18 12           CON(2)  (F.LSET)!(F.EXPR)
2506 E8E1A 3            CON(1)  3       OPCODE
2507 E8E1B 0480         CON(5)  #840
           0
2508 E8E20 00           CON(2)  0       NO VARIABLES
2509            *
2510            *
2511 E8E22 10           CON(2)  1       REGISTER TESTS
2512 E8E24 4            CON(1)  4       OPCODE MNEMONIC
2513 E8E25 F334         NIBASC  \?C<B  \
           C324
           0202
2514 E8E31 70           CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
2515 E8E33 3            CON(1)  3       OPCODE
2516 E8E34 1080         CON(5)  #801
           0
2517 E8E39 2            CON(1)  2       GROUP B
2518 E8E3A 0            CON(1)  0
2519            *
2520            *
2521 E8E3B 50           CON(2)  5       FIXED
2522 E8E3D 4            CON(1)  4       OPCODE MNEMONIC
```



```
2523 E8E3E 34D3         NIBASC  \C=R3  \
           2533
           0202
2524 E8E4A 50           CON(2)  (F.LSET)!(F.TREE)
2525 E8E4C 3            CON(1)  3       OPCODE
2526 E8E4D B110         CON(5)  #11B
           0
2527 E8E52 00           CON(2)  0       NO VARIABLES
2528            *
2529            *
2530 E8E54 50           CON(2)  5       FIXED
2531 E8E56 4            CON(1)  4       OPCODE MNEMONIC
2532 E8E57 3435         NIBASC  \CSRB  \
           2524
           0202
2533 E8E63 50           CON(2)  (F.LSET)!(F.TREE)
2534 E8E65 3            CON(1)  3       OPCODE
2535 E8E66 E180         CON(5)  #81E
           0
2536 E8E6B 00           CON(2)  0       NO VARIABLES
2537            *
2538            *
2539 E8E6D 50           CON(2)  5       FIXED
2540 E8E6F 6            CON(1)  6       OPCODE MNEMONIC
2541 E8E70 55E4         NIBASC  \UNCNFG\
           34E4
           6474
2542 E8E7C 50           CON(2)  (F.LSET)!(F.TREE)
2543 E8E7E 3            CON(1)  3       OPCODE
2544 E8E7F 4080         CON(5)  #804
           0
2545 E8E84 00           CON(2)  0       NO VARIABLES
```

```
2546                    STITLE  HASH TO GROUP 31
2547 E8E86      OPCG31
2548 E8E86 50           CON(2)  5       FIXED
2549 E8E88 5            CON(1)  5       OPCODE MNEMONIC
2550 E8E89 3444         NIBASC  \CDOXS \
           0385
           3502
2551 E8E95 10           CON(2)  (F.LSET)
2552 E8E97 3            CON(1)  3       OPCODE
2553 E8E98 E310         CON(5)  #13E
           0
2554 E8E9D 00           CON(2)  0       NO VARIABLES
2555            *
2556            *
2557 E8E9F 50           CON(2)  5       FIXED
2558 E8EA1 5            CON(1)  5       OPCODE MNEMONIC
2559 E8EA2 1425         NIBASC  \AR3EX \
           3354
           8502
2560 E8EAE 10           CON(2)  (F.LSET)
2561 E8EB0 3            CON(1)  3       OPCODE
2562 E8EB1 3210         CON(5)  #123
           0
2563 E8EB6 00           CON(2)  0       NO VARIABLES
2564            *
2565            *
2566 E8EB8 20           CON(2)  2       REGISTER ARITHMETIC
2567 E8EBA 3            CON(1)  3       OPCODE MNEMONIC
2568 E8EBB 3435         NIBASC  \CSR    \
           2502
           0202
2569 E8EC7 00           CON(2)  0
2570 E8EC9 3            CON(1)  3       OPCODE
2571 E8ECA 6000         CON(5)  #006
           0
2572 E8ECF 4            CON(1)  4       GROUP F
2573 E8ED0 0            CON(1)  0
2574            *
2575            *
2576 E8ED1 30           CON(2)  3       REGISTER LOGIC
2577 E8ED3 5            CON(1)  5       OPCODE MNEMONIC
2578 E8ED4 14D3         NIBASC  \A=A&C \
           1462
           3402
2579 E8EE0 10           CON(2)  (F.LSET)
2580 E8EE2 4            CON(1)  4       OPCODE
2581 E8EE3 60E0         CON(5)  #0E06
           0
2582 E8EE8 00           CON(2)  0       NO VARIABLES
2583            *
2584            *
2585 E8EEA 50           CON(2)  5       FIXED
2586 E8EEC 4            CON(1)  4       OPCODE MNEMONIC
2587 E8EED 34D3         NIBASC  \C=R1   \
           2513
```



```
           0202
2588 E8EF9 10           CON(2)  (F.LSET)
2589 E8EFB 3            CON(1)  3       OPCODE
2590 E8EFC 9110         CON(5)  #119
           0
2591 E8F01 00           CON(2)  0       NO VARIABLES
2592            *
2593            *
2594 E8F03 50           CON(2)  5       FIXED
2595 E8F05 5            CON(1)  5       OPCODE MNEMONIC
2596 E8F06 3425         NIBASC  \CR4EX \
           4354
           8502
2597 E8F12 50           CON(2)  (F.LSET)!(F.TREE)
2598 E8F14 3            CON(1)  3       OPCODE
2599 E8F15 C210         CON(5)  #12C
           0
2600 E8F1A 00           CON(2)  0       NO VARIABLES
2601            *
2602            *
2603 E8F1C 20           CON(2)  2       REGISTER ARITHMETIC
2604 E8F1E 5            CON(1)  5       OPCODE MNEMONIC
2605 E8F1F 44D3         NIBASC  \D=D-1 \
           44D2
           1302
2606 E8F2B 40           CON(2)  (F.TREE)
2607 E8F2D 3            CON(1)  3       OPCODE
2608 E8F2E F000         CON(5)  #00F
           0
2609 E8F33 1            CON(1)  1       GROUP C
2610 E8F34 0            CON(1)  0
2611            *
2612            *
2613 E8F35 10           CON(2)  1       REGISTER TESTS
2614 E8F37 5            CON(1)  5       OPCODE MNEMONIC
2615 E8F38 F334         NIBASC  \?C>=D \
           E3D3
           4402
2616 E8F44 70           CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
2617 E8F46 3            CON(1)  3       OPCODE
2618 E8F47 F080         CON(5)  #80F
           0
2619 E8F4C 2            CON(1)  2       GROUP B
2620 E8F4D 0            CON(1)  0
2621            *
2622            *
2623 E8F4E 30           CON(2)  3       REGISTER LOGIC
2624 E8F50 5            CON(1)  5       OPCODE MNEMONIC
2625 E8F51 14D3         NIBASC  \A=C!A \
           3412
           1402
2626 E8F5D 50           CON(2)  (F.LSET)!(F.TREE)
2627 E8F5F 4            CON(1)  4       OPCODE
2628 E8F60 E0E0         CON(5)  #0E0E
           0
```

```
2629 E8F65 00          CON(2)  0       NO VARIABLES
2630           *
2631           *
2632 E8F67 30          CON(2)  3       REGISTER LOGIC
2633 E8F69 5           CON(1)  5       OPCODE MNEMONIC
2634 E8F6A 34D3        NIBASC  \C=B&C \
           2462
           3402
2635 E8F76 50          CON(2)  (F.LSET)!(F.TREE)
2636 E8F78 4           CON(1)  4       OPCODE
2637 E8F79 50E0        CON(5)  #0E05
           0
2638 E8F7E 00          CON(2)  0       NO VARIABLES
2639           *
2640           *
2641 E8F80 00          CON(2)  0       NULL
2642 E8F82 0           CON(1)  0       OPCODE MNEMONIC
2643 E8F83 0202        NIBASC  \      \
           0202
           0202
2644 E8F8F 40          CON(2)  (F.TREE)
2645 E8F91 0           CON(1)  0       OPCODE
2646 E8F92 0000        CON(5)  #0
           0
2647 E8F97 00          CON(2)  0       NO VARIABLES
```



```
2648                   STITLE  HASH TO GROUP 32
2649 E8F99      OPCG32
2650 E8F99 20          CON(2)  2       REGISTER ARITHMETIC
2651 E8F9B 5           CON(1)  5       OPCODE MNEMONIC
2652 E8F9C 44D3        NIBASC  \D=D+1 \
           44B2
           1302
2653 E8FA8 00          CON(2)  0
2654 E8FAA 3           CON(1)  3       OPCODE
2655 E8FAB 7000        CON(5)  #007
           0
2656 E8FB0 3           CON(1)  3       GROUP E
2657 E8FB1 0           CON(1)  0
2658           *
2659           *
2660 E8FB2 C0          CON(2)  12      DATA POINTER ARITHMETIC
2661 E8FB4 6           CON(1)  6       OPCODE MNEMONIC
2662 E8FB5 4403        NIBASC  \D0=D0+\
           D344
           03B2
2663 E8FC1 12          CON(2)  (F.LSET)!(F.EXPR)
2664 E8FC3 3           CON(1)  3       OPCODE
2665 E8FC4 0610        CON(5)  #160
           0
2666 E8FC9 00          CON(2)  0       NO VARIABLES
2667           *
2668           *
2669 E8FCB 40          CON(2)  4       BRANCHES
2670 E8FCD 6           CON(1)  6       OPCODE MNEMONIC
2671 E8FCE 74F4        NIBASC  \GOSUBL\
           3555
           24C4
2672 E8FDA 16          CON(2)  (F.LSET)!(F.EXPR)!(F.GSUB)
2673 E8FDC 6           CON(1)  6       OPCODE
2674 E8FDD 000E        CON(5)  #8E000
           8
2675 E8FE2 3           CON(1)  3       FILL.MARK
2676 E8FE3 4           CON(1)  4       FILL.LENGTH
2677           *
2678           *
2679 E8FE4 50          CON(2)  5       FIXED
2680 E8FE6 5           CON(1)  5       OPCODE MNEMONIC
2681 E8FE7 4403        NIBASC  \D0=CS \
           D334
           3502
2682 E8FF3 10          CON(2)  (F.LSET)
2683 E8FF5 3           CON(1)  3       OPCODE
2684 E8FF6 C310        CON(5)  #13C
           0
2685 E8FFB 00          CON(2)  0       NO VARIABLES
2686           *
2687           *
2688 E8FFD 40          CON(2)  4       BRANCHES
2689 E8FFF 6           CON(1)  6       OPCODE MNEMONIC
2690 E9000 4413        NIBASC  \D1=(4)\
```

```
            D382
            4392
2691 E900C 5B          CON(2)  (F.LSET)!(F.ABSL)!(F.TREE)!(F.EXPR)!(F.PART)
2692 E900E 6           CON(1)  6       OPCODE
2693 E900F 000E        CON(5)  #1E000
            1
2694 E9014 3           CON(1)  3       FILL.MARK
2695 E9015 4           CON(1)  4       FILL.LENGTH
2696            *
2697            *
2698 E9016 50          CON(2)  5       FIXED
2699 E9018 4           CON(1)  4       OPCODE MNEMONIC
2700 E901A 4435        NIBASC  \DSRB  \
            2524
            0202
2701 E9025 50          CON(2)  (F.LSET)!(F.TREE)
2702 E9027 3           CON(1)  3       OPCODE
2703 E9028 F180        CON(5)  #81F
            0
2704 E902D 00          CON(2)  0       NO VARIABLES
2705            *
2706            *
2707 E902F 50          CON(2)  5       FIXED
2708 E9031 5           CON(1)  5       OPCODE MNEMONIC
2709 E9032 94E4        NIBASC  \INTON \
            45F4
            E402
2710 E903E 50          CON(2)  (F.LSET)!(F.TREE)
2711 E9040 4           CON(1)  4       OPCODE
2712 E9041 0808        CON(5)  #8080
            0
2713 E9046 00          CON(2)  0       NO VARIABLES
2714            *
2715            *
2716 E9048 10          CON(2)  1       REGISTER TESTS
2717 E904A 4           CON(1)  4       OPCODE MNEMONIC
2718 E904B F324        NIBASC  \?B>C  \
            E334
            0202
2719 E9057 70          CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
2720 E9059 3           CON(1)  3       OPCODE
2721 E905A 1080        CON(5)  #801
            0
2722 E905F 2           CON(1)  2       GROUP B
2723 E9060 0           CON(1)  0
2724            *
2725            *
2726 E9061 00          CON(2)  0       NULL
2727 E9063 0           CON(1)  0       OPCODE MNEMONIC
2728 E9064 0202        NIBASC  \      \
            0202
            0202
2729 E9070 40          CON(2)  (F.TREE)
2730 E9072 0           CON(1)  0       OPCODE
2731 E9073 0000        CON(5)  #0
```



```
            0
2732 E9078 00          CON(2)  0       NO VARIABLES
```

```
2733                    STITLE  HASH TO GROUP 33
2734 E907A      OPCG33
2735 E907A 50           CON(2)  5       FIXED
2736 E907C 5            CON(1)  5       OPCODE MNEMONIC
2737 E907D 4403         NIBASC  \D0=AS \
           D314
           3502
2738 E9089 10           CON(2)  (F.LSET)
2739 E908B 3            CON(1)  3       OPCODE
2740 E908C 8310         CON(5)  #138
           0
2741 E9091 00           CON(2)  0       NO VARIABLES
2742            *
2743            *
2744 E9093 20           CON(2)  2       REGISTER ARITHMETIC
2745 E9095 5            CON(1)  5       OPCODE MNEMONIC
2746 E9096 14D3         NIBASC  \A=B-A \
           24D2
           1402
2747 E90A2 00           CON(2)  0
2748 E90A4 3            CON(1)  3       OPCODE
2749 E90A5 C000         CON(5)  #00C
           0
2750 E90AA 3            CON(1)  3       GROUP E
2751 E90AB 0            CON(1)  0
2752            *
2753            *
2754 E90AC 50           CON(2)  5       FIXED
2755 E90AE 4            CON(1)  4       OPCODE MNEMONIC
2756 E90AF D405         NIBASC  \MP=0  \
           D303
           0202
2757 E90BB 10           CON(2)  (F.LSET)
2758 E90BD 3            CON(1)  3       OPCODE
2759 E90BE 8280         CON(5)  #828
           0
2760 E90C3 00           CON(2)  0       NO VARIABLES
2761            *
2762            *
2763 E90C5 10           CON(2)  1       REGISTER TESTS
2764 E90C7 4            CON(1)  4       OPCODE MNEMONIC
2765 E90C8 F324         NIBASC  \?B>A  \
           E314
           0202
2766 E90D4 30           CON(2)  (F.LSET)!(F.RETY)
2767 E90D6 3            CON(1)  3       OPCODE
2768 E90D7 4080         CON(5)  #804
           0
2769 E90DC 2            CON(1)  2       GROUP B
2770 E90DD 0            CON(1)  0
2771            *
2772            *
2773 E90DE 20           CON(2)  2       REGISTER ARITHMETIC
2774 E90E0 5            CON(1)  5       OPCODE MNEMONIC
2775 E90E1 34D3         NIBASC  \C=C-A \
```



```
           34D2
           1402
2776 E90ED 00           CON(2)  0
2777 E90EF 3            CON(1)  3       OPCODE
2778 E90F0 2000         CON(5)  #002
           0
2779 E90F5 3            CON(1)  3       GROUP E
2780 E90F6 0            CON(1)  0
2781            *
2782            *
2783 E90F7 20           CON(2)  2       REGISTER ARITHMETIC
2784 E90F9 3            CON(1)  3       OPCODE MNEMONIC
2785 E90FA 4435         NIBASC  \DSR   \
           2502
           0202
2786 E9106 40           CON(2)  (F.TREE)
2787 E9108 3            CON(1)  3       OPCODE
2788 E9109 7000         CON(5)  #007
           0
2789 E910E 4            CON(1)  4       GROUP F
2790 E910F 0            CON(1)  0
2791            *
2792            *
2793 E9110 40           CON(2)  4       BRANCHES
2794 E9112 6            CON(1)  6       OPCODE MNEMONIC
2795 E9113 2554         NIBASC  \REL(4)\
           C482
           4392
2796 E911F 52           CON(2)  (F.LSET)!(F.TREE)!(F.EXPR)
2797 E9121 4            CON(1)  4       OPCODE
2798 E9122 0000         CON(5)  #0
           0
2799 E9127 1            CON(1)  1       FILL.MARK
2800 E9128 4            CON(1)  4       FILL.LENGTH
2801            *
2802            *
2803 E9129 10           CON(2)  1       REGISTER TESTS
2804 E912B 4            CON(1)  4       OPCODE MNEMONIC
2805 E912C F314         NIBASC  \?A#C  \
           3234
           0202
2806 E9138 70           CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
2807 E913A 3            CON(1)  3       OPCODE
2808 E913B 6080         CON(5)  #806
           0
2809 E9140 1            CON(1)  1       GROUP A
2810 E9141 0            CON(1)  0
2811            *
2812            *
2813 E9142 10           CON(2)  1       REGISTER TESTS
2814 E9144 4            CON(1)  4       OPCODE MNEMONIC
2815 E9145 F334         NIBASC  \?C#D  \
           3244
           0202
2816 E9151 70           CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
```

```
2817 E9153 3           CON(1)  3       OPCODE
2818 E9154 7080        CON(5)  #807
           0
2819 E9159 1           CON(1)  1       GROUP A
2820 E915A 0           CON(1)  0
2821           *
2822           *
2823 E915B 20          CON(2)  2       REGISTER ARITHMETIC
2824 E915D 5           CON(1)  5       OPCODE MNEMONIC
2825 E915E 34D3        NIBASC  \C=C-1 \
           34D2
           1302
2826 E916A 40          CON(2)  (F.TREE)
2827 E916C 3           CON(1)  3       OPCODE
2828 E916D E000        CON(5)  #00E
           0
2829 E9172 1           CON(1)  1       GROUP C
2830 E9173 0           CON(1)  0
2831           *
2832           *
2833 E9174 40          CON(2)  4       BRANCHES
2834 E9176 6           CON(1)  6       OPCODE MNEMONIC
2835 E9177 34F4        NIBASC  \CON(2)\
           E482
           2392
2836 E9183 53          CON(2)  (F.LSET)!(F.TREE)!(F.ABSL)!(F.EXPR)
2837 E9185 2           CON(1)  2       OPCODE
2838 E9186 0000        CON(5)  #0
           0
2839 E918B 1           CON(1)  1       FILL.MARK
2840 E918C 2           CON(1)  2       FILL.LENGTH
```



```
2841                   STITLE  HASH TO GROUP 34
2842 E918D     OPCG34
2843 E918D 20          CON(2)  2       REGISTER ARITHMETIC
2844 E918F 5           CON(1)  5       OPCODE MNEMONIC
2845 E9190 34D3        NIBASC  \C=C+1 \
           34B2
           1302
2846 E919C 00          CON(2)  0
2847 E919E 3           CON(1)  3       OPCODE
2848 E919F 6000        CON(5)  #006
           0
2849 E91A4 3           CON(1)  3       GROUP E
2850 E91A5 0           CON(1)  0
2851           *
2852           *
2853 E91A6 20          CON(2)  2       REGISTER ARITHMETIC
2854 E91A8 5           CON(1)  5       OPCODE MNEMONIC
2855 E91A9 14D3        NIBASC  \A=B+A \
           24B2
           1402
2856 E91B5 00          CON(2)  0
2857 E91B7 3           CON(1)  3       OPCODE
2858 E91B8 0000        CON(5)  #000
           0
2859 E91BD 1           CON(1)  1       GROUP A
2860 E91BE 0           CON(1)  0
2861           *
2862           *
2863 E91BF C0          CON(2)  12      DATA POINTER ARITHMETIC
2864 E91C1 6           CON(1)  6       OPCODE MNEMONIC
2865 E91C2 4403        NIBASC  \D0=D0-\
           D344
           03D2
2866 E91CE 12          CON(2)  (F.LSET)!(F.EXPR)
2867 E91D0 3           CON(1)  3       OPCODE
2868 E91D1 0810        CON(5)  #180
           0
2869 E91D6 00          CON(2)  0       NO VARIABLES
2870           *
2871           *
2872 E91D8 10          CON(2)  1       REGISTER TESTS
2873 E91DA 5           CON(1)  5       OPCODE MNEMONIC
2874 E91DB F344        NIBASC  \?D>=C \
           E3D3
           3402
2875 E91E7 30          CON(2)  (F.LSET)!(F.RETY)
2876 E91E9 3           CON(1)  3       OPCODE
2877 E91EA B080        CON(5)  #80B
           0
2878 E91EF 2           CON(1)  2       GROUP B
2879 E91F0 0           CON(1)  0
2880           *
2881           *
2882 E91F1 42          CON(2)  36      PSEUDO-OP
2883 E91F3 3           CON(1)  3       OPCODE MNEMONIC
```

```
2884 E91F4 2494          NIBASC  \BIN   \
           E402
           0202
2885 E9200 42            CON(2)  (F.TREE)!(F.EXPR)
2886 E9202 0             CON(1)  0       OPCODE
2887 E9203 0000          CON(5)  #0
           0
2888 E9208 00            CON(2)  0       NO VARIABLES
2889             *
2890             *
2891 E920A 20            CON(2)  2       REGISTER ARITHMETIC
2892 E920C 5             CON(1)  5       OPCODE MNEMONIC
2893 E920D 34D3          NIBASC  \C=C+A \
           34B2
           1402
2894 E9219 40            CON(2)  (F.TREE)
2895 E921B 3             CON(1)  3       OPCODE
2896 E921C 2000          CON(5)  #002
           0
2897 E9221 1             CON(1)  1       GROUP C
2898 E9222 0             CON(1)  0
2899             *
2900             *
2901 E9223 40            CON(2)  4       BRANCHES
2902 E9225 6             CON(1)  6       OPCODE MNEMONIC
2903 E9226 74F4          NIBASC  \GOVLNG\
           65C4
           E474
2904 E9232 53            CON(2)  (F.LSET)!(F.TREE)!(F.ABSL)!(F.EXPR)
2905 E9234 7             CON(1)  7       OPCODE
2906 E9235 000D          CON(5)  #8D000
           8
2907 E923A 3             CON(1)  3       FILL.MARK
2908 E923B 5             CON(1)  5       FILL.LENGTH
2909             *
2910             *
2911 E923C 10            CON(2)  1       REGISTER TESTS
2912 E923E 4             CON(1)  4       OPCODE MNEMONIC
2913 E923F F334          NIBASC  \?C#B  \
           3224
           0202
2914 E924B 70            CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
2915 E924D 3             CON(1)  3       OPCODE
2916 E924E 5080          CON(5)  #805
           0
2917 E9253 1             CON(1)  1       GROUP A
2918 E9254 0             CON(1)  0
2919             *
2920             *
2921 E9255 00            CON(2)  0       NULL
2922 E9257 0             CON(1)  0       OPCODE MNEMONIC
2923 E9258 0202          NIBASC  \      \
           0202
           0202
2924 E9264 40            CON(2)  (F.TREE)
```



```
2925 E9266 0             CON(1)  0       OPCODE
2926 E9267 0000          CON(5)  #0
           0
2927 E926C 00            CON(2)  0       NO VARIABLES
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER

```
2928                    STITLE  HASH TO GROUP 35
2929 E926E      OPCG35
2930 E926E 30           CON(2)  3       REGISTER LOGIC
2931 E9270 5            CON(1)  5       OPCODE MNEMONIC
2932 E9271 34D3         NIBASC  \C=C!A \
           3412
           1402
2933 E927D 10           CON(2)  (F.LSET)
2934 E927F 4            CON(1)  4       OPCODE
2935 E9280 A0E0         CON(5)  #0E0A
           0
2936 E9285 00           CON(2)  0       NO VARIABLES
2937            *
2938            *
2939 E9287 20           CON(2)  2       REGISTER ARITHMETIC
2940 E9289 5            CON(1)  5       OPCODE MNEMONIC
2941 E928A 24D3         NIBASC  \B=B-1 \
           24D2
           1302
2942 E9296 00           CON(2)  0
2943 E9298 3            CON(1)  3       OPCODE
2944 E9299 D000         CON(5)  #00D
           0
2945 E929E 1            CON(1)  1       GROUP C
2946 E929F 0            CON(1)  0
2947            *
2948            *
2949 E92A0 50           CON(2)  5       FIXED
2950 E92A2 5            CON(1)  5       OPCODE MNEMONIC
2951 E92A3 3425         NIBASC  \CR3EX \
           3354
           8502
2952 E92AF 10           CON(2)  (F.LSET)
2953 E92B1 3            CON(1)  3       OPCODE
2954 E92B2 B210         CON(5)  #12B
           0
2955 E92B7 00           CON(2)  0       NO VARIABLES
2956            *
2957            *
2958 E92B9 30           CON(2)  3       REGISTER LOGIC
2959 E92BB 5            CON(1)  5       OPCODE MNEMONIC
2960 E92BC 14D3         NIBASC  \A=B!A \
           2412
           1402
2961 E92C8 10           CON(2)  (F.LSET)
2962 E92CA 4            CON(1)  4       OPCODE
2963 E92CB 80E0         CON(5)  #0E08
           0
2964 E92D0 00           CON(2)  0       NO VARIABLES
2965            *
2966            *
2967 E92D2 30           CON(2)  3       REGISTER LOGIC
2968 E92D4 5            CON(1)  5       OPCODE MNEMONIC
2969 E92D5 34D3         NIBASC  \C=A&C \
           1462
```



```
           3402
2970 E92E1 10           CON(2)  (F.LSET)
2971 E92E3 4            CON(1)  4       OPCODE
2972 E92E4 20E0         CON(5)  #0E02
           0
2973 E92E9 00           CON(2)  0       NO VARIABLES
2974            *
2975            *
2976 E92EB 30           CON(2)  3       REGISTER LOGIC
2977 E92ED 5            CON(1)  5       OPCODE MNEMONIC
2978 E92EE 34D3         NIBASC  \C=C&D \
           3462
           4402
2979 E92FA 50           CON(2)  (F.LSET)!(F.TREE)
2980 E92FC 4            CON(1)  4       OPCODE
2981 E92FD 70E0         CON(5)  #0E07
           0
2982 E9302 00           CON(2)  0       NO VARIABLES
2983            *
2984            *
2985 E9304 50           CON(2)  5       FIXED
2986 E9306 4            CON(1)  4       OPCODE MNEMONIC
2987 E9307 85D4         NIBASC  \XM=0  \
           D303
           0202
2988 E9313 50           CON(2)  (F.LSET)!(F.TREE)
2989 E9315 3            CON(1)  3       OPCODE
2990 E9316 1280         CON(5)  #821
           0
2991 E931B 00           CON(2)  0       NO VARIABLES
2992            *
2993            *
2994 E931D 10           CON(2)  1       REGISTER TESTS
2995 E931F 4            CON(1)  4       OPCODE MNEMONIC
2996 E9320 F334         NIBASC  \?C#0  \
           3203
           0202
2997 E932C 70           CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
2998 E932E 3            CON(1)  3       OPCODE
2999 E932F E080         CON(5)  #80E
           0
3000 E9334 1            CON(1)  1       GROUP A]
3001 E9335 0            CON(1)  0
3002            *
3003            *
3004 E9336 50           CON(2)  5       FIXED
3005 E9338 5            CON(1)  5       OPCODE MNEMONIC
3006 E9339 1425         NIBASC  \AR2EX \
           2354
           8502
3007 E9345 50           CON(2)  (F.LSET)!(F.TREE)
3008 E9347 3            CON(1)  3       OPCODE
3009 E9348 2210         CON(5)  #122
           0
3010 E934D 00           CON(2)  0       NO VARIABLES
```

```
3011                *
3012                *
3013 E934F 20              CON(2)  2       REGISTER ARITHMETIC
3014 E9351 5               CON(1)  5       OPCODE MNEMONIC
3015 E9352 24D3            NIBASC  \B=B-A \
           24D2
           1402
3016 E935E 40              CON(2)  (F.TREE)
3017 E9360 3               CON(1)  3       OPCODE
3018 E9361 8000            CON(5)  #008
           0
3019 E9366 3               CON(1)  3       GROUP E
3020 E9367 0               CON(1)  0
3021                *
3022                *
3023 E9368 00              CON(2)  0       NULL
3024 E936A 0               CON(1)  0       OPCODE MNEMONIC
3025 E936B 0202            NIBASC  \       \
           0202
           0202
3026 E9377 40              CON(2)  (F.TREE)
3027 E9379 0               CON(1)  0       OPCODE
3028 E937A 0000            CON(5)  #0
           0
3029 E937F 00              CON(2)  0       NO VARIABLES
```



```
3030                    STITLE  HASH TO GROUP 36
3031 E9381      OPCG36
3032 E9381 20              CON(2)  2       REGISTER ARITHMETIC
3033 E9383 5               CON(1)  5       OPCODE MNEMONIC
3034 E9384 24D3            NIBASC  \B=B+1 \
           24B2
           1302
3035 E9390 00              CON(2)  0
3036 E9392 3               CON(1)  3       OPCODE
3037 E9393 5000            CON(5)  #005
           0
3038 E9398 3               CON(1)  3       GROUP E
3039 E9399 0               CON(1)  0
3040                *
3041                *
3042 E939A 10              CON(2)  1       REGISTER TESTS
3043 E939C 4               CON(1)  4       OPCODE MNEMONIC
3044 E939D F324            NIBASC  \?B=C  \
           D334
           0202
3045 E93A9 30              CON(2)  (F.LSET)!(F.RETY)
3046 E93AB 3               CON(1)  3       OPCODE
3047 E93AC 1080            CON(5)  #801
           0
3048 E93B1 1               CON(1)  1       GROUP A
3049 E93B2 0               CON(1)  0
3050                *
3051                *
3052 E93B3 52              CON(2)  37      FORTH
3053 E93B5 5               CON(1)  5       OPCODE MNEMONIC
3054 E93B6 64F4            NIBASC  \FORTH \
           2545
           8402
3055 E93C2 00              CON(2)  0
3056 E93C4 0               CON(1)  0       OPCODE
3057 E93C5 0000            CON(5)  #0
           0
3058 E93CA 00              CON(2)  0       NO VARIABLES
3059                *
3060                *
3061 E93CC 10              CON(2)  1       REGISTER TESTS
3062 E93CE 5               CON(1)  5       OPCODE MNEMONIC
3063 E93CF F314            NIBASC  \?A>=B \
           E3D3
           2402
3064 E93DB 70              CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
3065 E93DD 3               CON(1)  3       OPCODE
3066 E93DE 8080            CON(5)  #808
           0
3067 E93E3 2               CON(1)  2       GROUP B
3068 E93E4 0               CON(1)  0
3069                *
3070                *
3071 E93E5 20              CON(2)  2       REGISTER ARITHMETIC
3072 E93E7 6               CON(1)  6       OPCODE MNEMONIC
```

```
3073 E93E8 24D3        NIBASC  \B=-B-1\
           D224
           D213
3074 E93F4 40          CON(2)  (F.TREE)
3075 E93F6 3           CON(1)  3       OPCODE
3076 E93F7 D000        CON(5)  #00D
           0
3077 E93FC 4           CON(1)  4       GROUP F
3078 E93FD 0           CON(1)  0
3079             *
3080             *
3081 E93FE 20          CON(2)  2       REGISTER ARITHMETIC
3082 E9400 5           CON(1)  5       OPCODE MNEMONIC
3083 E9401 24D3        NIBASC  \B=B+A \
           24B2
           1402
3084 E940D 40          CON(2)  (F.TREE)
3085 E940F 3           CON(1)  3       OPCODE
3086 E9410 8000        CON(5)  #008
           0
3087 E9415 1           CON(1)  1       GROUP C
3088 E9416 0           CON(1)  0
3089             *
3090             *
3091 E9417 40          CON(2)  4       BRANCHES
3092 E9419 3           CON(1)  3       OPCODE MNEMONIC
3093 E941A 74F4        NIBASC  \GOC   \
           3402
           0202
3094 E9426 52          CON(2)  (F.LSET)!(F.TREE)!(F.EXPR)
3095 E9428 3           CON(1)  3       OPCODE
3096 E9429 0040        CON(5)  #400
           0
3097 E942E 2           CON(1)  2       FILL.MARK
3098 E942F 2           CON(1)  2       FILL.LENGTH
```



```
3099                   STITLE  HASH TO GROUP 37
3100 E9430      OPCG37
3101 E9430 30          CON(2)  3       REGISTER LOGIC
3102 E9432 5           CON(1)  5       OPCODE MNEMONIC
3103 E9433 24D3        NIBASC  \B=B!A \
           2412
           1402
3104 E943F 10          CON(2)  (F.LSET)
3105 E9441 4           CON(1)  4       OPCODE
3106 E9442 C0E0        CON(5)  #0E0C
           0
3107 E9447 00          CON(2)  0       NO VARIABLES
3108             *
3109             *
3110 E9449 80          CON(2)  8       POINTER TESTS
3111 E944B 3           CON(1)  3       OPCODE MNEMONIC
3112 E944C F305        NIBASC  \?P=   \
           D302
           0202
3113 E9458 32          CON(2)  (F.LSET)!(F.RETY)!(F.EXPR)
3114 E945A 3           CON(1)  3       OPCODE
3115 E945B 0980        CON(5)  #890
           0
3116 E9460 00          CON(2)  0       NO VARIABLES
3117             *
3118             *
3119 E9462 50          CON(2)  5       FIXED
3120 E9464 4           CON(1)  4       OPCODE MNEMONIC
3121 E9465 2533        NIBASC  \R3=C  \
           D334
           0202
3122 E9471 10          CON(2)  (F.LSET)
3123 E9473 3           CON(1)  3       OPCODE
3124 E9474 B010        CON(5)  #10B
           0
3125 E9479 00          CON(2)  0       NO VARIABLES
3126             *
3127             *
3128 E947B 10          CON(2)  1       REGISTER TESTS
3129 E947D 4           CON(1)  4       OPCODE MNEMONIC
3130 E947E F324        NIBASC  \?B=A  \
           D314
           0202
3131 E948A 70          CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
3132 E948C 3           CON(1)  3       OPCODE
3133 E948D 0080        CON(5)  #800
           0
3134 E9492 1           CON(1)  1       GROUP A
3135 E9493 0           CON(1)  0
3136             *
3137             *
3138 E9494 20          CON(2)  2       REGISTER ARITHMETIC
3139 E9496 5           CON(1)  5       OPCODE MNEMONIC
3140 E9497 14D3        NIBASC  \A=A-1 \
           14D2
```

```
            1302
3141 E94A3 40          CON(2)  (F.TREE)
3142 E94A5 3           CON(1)  3       OPCODE
3143 E94A6 C000        CON(5)  #00C
           0
3144 E94AB 1           CON(1)  1       GROUP C
3145 E94AC 0           CON(1)  0
3146           *
3147           *
3148 E94AD 30          CON(2)  3       REGISTER LOGIC
3149 E94AF 5           CON(1)  5       OPCODE MNEMONIC
3150 E94B0 44D3        NIBASC  \D=C&D \
           3462
           4402
3151 E94BC 50          CON(2)  (F.LSET)!(F.TREE)
3152 E94BE 4           CON(1)  4       OPCODE
3153 E94BF 30E0        CON(5)  #0E03
           0
3154 E94C4 00          CON(2)  0       NO VARIABLES
3155           *
3156           *
3157 E94C6 00          CON(2)  0       NULL
3158 E94C8 0           CON(1)  0       OPCODE MNEMONIC
3159 E94C9 0202        NIBASC  \      \
           0202
           0202
3160 E94D5 40          CON(2)  (F.TREE)
3161 E94D7 0           CON(1)  0       OPCODE
3162 E94D8 0000        CON(5)  #0
           0
3163 E94DD 00          CON(2)  0       NO VARIABLES
```



```
3164                   STITLE  HASH TO GROUP 38
3165 E94DF     OPCG38
3166 E94DF 21          CON(2)  18      LCASC
3167 E94E1 5           CON(1)  5       OPCODE MNEMONIC
3168 E94E2 C434        NIBASC  \LCASC \
           1435
           3402
3169 E94EE 02          CON(2)  (F.EXPR)
3170 E94F0 2           CON(1)  2       OPCODE
3171 E94F1 0300        CON(5)  #30
           0
3172 E94F6 3           CON(1)  3       FILL.MARK
3173 E94F7 2           CON(1)  2       FILL.LENGTH
3174           *
3175           *
3176 E94F8 20          CON(2)  2       REGISTER ARITHMETIC
3177 E94FA 4           CON(1)  4       OPCODE MNEMONIC
3178 E94FB 24D3        NIBASC  \B=-B  \
           D224
           0202
3179 E9507 00          CON(2)  0
3180 E9509 3           CON(1)  3       OPCODE
3181 E950A 9000        CON(5)  #009
           0
3182 E950F 4           CON(1)  4       GROUP F
3183 E9510 0           CON(1)  0
3184           *
3185           *
3186 E9511 91          CON(2)  25      STITLE PSEUDO-OP
3187 E9513 6           CON(1)  6       OPCODE MNEMONIC
3188 E9514 3545        NIBASC  \STITLE\
           9445
           C454
3189 E9520 10          CON(2)  (F.LSET)
3190 E9522 0           CON(1)  0       OPCODE
3191 E9523 0000        CON(5)  #0
           0
3192 E9528 00          CON(2)  0       NO VARIABLES
3193           *
3194           *
3195 E952A 20          CON(2)  2       REGISTER ARITHMETIC
3196 E952C 5           CON(1)  5       OPCODE MNEMONIC
3197 E952D 14D3        NIBASC  \A=A+1 \
           14B2
           1302
3198 E9539 00          CON(2)  0
3199 E953B 3           CON(1)  3       OPCODE
3200 E953C 4000        CON(5)  #004
           0
3201 E9541 3           CON(1)  3       GROUP E
3202 E9542 0           CON(1)  0
3203           *
3204           *
3205 E9543 50          CON(2)  5       FIXED
3206 E9545 6           CON(1)  6       OPCODE MNEMONIC
```

```
3207 E9546 94E4          NIBASC  \INTOFF\
           45F4
           6464
3208 E9552 10            CON(2)  (F.LSET)
3209 E9554 4             CON(1)  4       OPCODE
3210 E9555 F808          CON(5)  #808F
           0
3211 E955A 00            CON(2)  0       NO VARIABLES
3212             *
3213             *
3214 E955C 50            CON(2)  5       FIXED
3215 E955E 4             CON(1)  4       OPCODE MNEMONIC
3216 E955F 2533          NIBASC  \R3=A  \
           D314
           0202
3217 E956B 50            CON(2)  (F.LSET)!(F.TREE)
3218 E956D 3             CON(1)  3       OPCODE
3219 E956E 3010          CON(5)  #103
           0
3220 E9573 00            CON(2)  0       NO VARIABLES
3221             *
3222             *
3223 E9575 A1            CON(2)  26      FORTH WORD
3224 E9577 4             CON(1)  4       OPCODE MNEMONIC
3225 E9578 75F4          NIBASC  \WORD  \
           2544
           0202
3226 E9584 42            CON(2)  (F.TREE)!(F.EXPR)
3227 E9586 0             CON(1)  0       OPCODE
3228 E9587 0000          CON(5)  #0
           0
3229 E958C 00            CON(2)  0       NO VARIABLES
3230             *
3231             *
3232 E958E 10            CON(2)  1       REGISTER TESTS
3233 E9590 4             CON(1)  4       OPCODE MNEMONIC
3234 E9591 F344          NIBASC  \?D=0  \
           D303
           0202
3235 E959D 70            CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
3236 E959F 3             CON(1)  3       OPCODE
3237 E95A0 B080          CON(5)  #80B
           0
3238 E95A5 1             CON(1)  1       GROUP A
3239 E95A6 0             CON(1)  0
3240             *
3241             *
3242 E95A7 20            CON(2)  2       REGISTER ARITHMETIC
3243 E95A9 5             CON(1)  5       OPCODE MNEMONIC
3244 E95AA 14D3          NIBASC  \A=A+A \
           14B2
           1402
3245 E95B6 40            CON(2)  (F.TREE)
3246 E95B8 3             CON(1)  3       OPCODE
3247 E95B9 4000          CON(5)  #004
```



```
           0
3248 E95BE 1             CON(1)  1       GROUP C
3249 E95BF 0             CON(1)  0
3250             *
3251             *
3252 E95C0 20            CON(2)  2       REGISTER ARITHMETIC
3253 E95C2 4             CON(1)  4       OPCODE MNEMONIC
3254 E95C3 2414          NIBASC  \BAEX  \
           5485
           0202
3255 E95CF 40            CON(2)  (F.TREE)
3256 E95D1 3             CON(1)  3       OPCODE
3257 E95D2 C000          CON(5)  #00C
           0
3258 E95D7 2             CON(1)  2       GROUP D
3259 E95D8 0             CON(1)  0
3260             *
3261             *
3262 E95D9 00            CON(2)  0       NULL
3263 E95DB 0             CON(1)  0       OPCODE MNEMONIC
3264 E95DC 0202          NIBASC  \      \
           0202
           0202
3265 E95E8 40            CON(2)  (F.TREE)
3266 E95EA 0             CON(1)  0       OPCODE
3267 E95EB 0000          CON(5)  #0
           0
3268 E95F0 00            CON(2)  0       NO VARIABLES
```

```
3269                    STITLE  HASH TO GROUP 39
3270 E95F2      OPCG39
3271 E95F2 50           CON(2)  5       FIXED
3272 E95F4 5            CON(1)  5       OPCODE MNEMONIC
3273 E95F5 3425         NIBASC  \CR2EX \
           2354
           8502
3274 E9601 10           CON(2)  (F.LSET)
3275 E9603 3            CON(1)  3       OPCODE
3276 E9604 A210         CON(5)  #12A
           0
3277 E9609 00           CON(2)  0       NO VARIABLES
3278             *
3279             *
3280 E960B 50           CON(2)  5       FIXED
3281 E960D 4            CON(1)  4       OPCODE MNEMONIC
3282 E960E 1435         NIBASC  \ASLC  \
           C434
           0202
3283 E961A 10           CON(2)  (F.LSET)
3284 E961C 3            CON(1)  3       OPCODE
3285 E961D 0180         CON(5)  #810
           0
3286 E9622 00           CON(2)  0       NO VARIABLES
3287             *
3288             *
3289 E9624 20           CON(2)  2       REGISTER ARITHMETIC
3290 E9626 6            CON(1)  6       OPCODE MNEMONIC
3291 E9627 44D3         NIBASC  \D=-D-1\
           D244
           D213
3292 E9633 00           CON(2)  0
3293 E9635 3            CON(1)  3       OPCODE
3294 E9636 F000         CON(5)  #00F
           0
3295 E963B 4            CON(1)  4       GROUP F
3296 E963C 0            CON(1)  0
3297             *
3298             *
3299 E963D 10           CON(2)  1       REGISTER TESTS
3300 E963F 5            CON(1)  5       OPCODE MNEMONIC
3301 E9640 F334         NIBASC  \?C<=D \
           C3D3
           4402
3302 E964C 30           CON(2)  (F.LSET)!(F.RETY)
3303 E964E 3            CON(1)  3       OPCODE
3304 E964F B080         CON(5)  #80B
           0
3305 E9654 2            CON(1)  2       GROUP B
3306 E9655 0            CON(1)  0
3307             *
3308             *
3309 E9656 20           CON(2)  2       REGISTER ARITHMETIC
3310 E9658 5            CON(1)  5       OPCODE MNEMONIC
3311 E9659 24D3         NIBASC  \B=C-B \
```



```
           34D2
           2402
3312 E9665 40           CON(2)  (F.TREE)
3313 E9667 3            CON(1)  3       OPCODE
3314 E9668 D000         CON(5)  #00D
           0
3315 E966D 3            CON(1)  3       GROUP E
3316 E966E 0            CON(1)  0
3317             *
3318             *
3319 E966F 01           CON(2)  16      Dx=HEX INSTRUCTIONS
3320 E9671 6            CON(1)  6       OPCODE MNEMONIC
3321 E9672 4413         NIBASC  \D1=HEX\
           D384
           5485
3322 E967E 50           CON(2)  (F.LSET)!(F.TREE)
3323 E9680 4            CON(1)  4       OPCODE
3324 E9681 00D1         CON(5)  #1D00
           0
3325 E9686 3            CON(1)  3       FILL.MARK
3326 E9687 2            CON(1)  2       FILL.LENGTH
3327             *
3328             *
3329 E9688 50           CON(2)  5       FIXED
3330 E968A 5            CON(1)  5       OPCODE MNEMONIC
3331 E968B 05D3         NIBASC  \P=P-1 \
           05D2
           1302
3332 E9697 50           CON(2)  (F.LSET)!(F.TREE)
3333 E9699 2            CON(1)  2       OPCODE
3334 E969A D000         CON(5)  #0D
           0
3335 E969F 00           CON(2)  0       NO VARIABLES
3336             *
3337             *
3338 E96A1 10           CON(2)  1       REGISTER TESTS
3339 E96A3 5            CON(1)  5       OPCODE MNEMONIC
3340 E96A4 F324         NIBASC  \?B>=A \
           E3D3
           1402
3341 E96B0 70           CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
3342 E96B2 3            CON(1)  3       OPCODE
3343 E96B3 C080         CON(5)  #80C
           0
3344 E96B8 2            CON(1)  2       GROUP B
3345 E96B9 0            CON(1)  0
3346             *
3347             *
3348 E96BA 50           CON(2)  5       FIXED
3349 E96BC 5            CON(1)  5       OPCODE MNEMONIC
3350 E96BD 1425         NIBASC  \AR1EX \
           1354
           8502
3351 E96C9 50           CON(2)  (F.LSET)!(F.TREE)
3352 E96CB 3            CON(1)  3       OPCODE
```

```
3353 E96CC 1210          CON(5)  #121
           0
3354 E96D1 00            CON(2)  0       NO VARIABLES
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER


```
3355                     STITLE  HASH TO GROUP 40
3356 E96D3       OPCG40
3357 E96D3 40            CON(2)  4       BRANCHES
3358 E96D5 6             CON(1)  6       OPCODE MNEMONIC
3359 E96D6 4413          NIBASC  \D1=(5)\
           D382
           5392
3360 E96E2 1B            CON(2)  (F.LSET)!(F.ABSL)!(F.EXPR)!(F.PART)
3361 E96E4 7             CON(1)  7       OPCODE
3362 E96E5 000F          CON(5)  #1F000
           1
3363 E96EA 3             CON(1)  3       FILL.MARK
3364 E96EB 5             CON(1)  5       FILL.LENGTH
3365               *
3366               *
3367 E96EC 20            CON(2)  2       REGISTER ARITHMETIC
3368 E96EE 5             CON(1)  5       OPCODE MNEMONIC
3369 E96EF 24D3          NIBASC  \B=C+B \
           34B2
           2402
3370 E96FB 00            CON(2)  0
3371 E96FD 3             CON(1)  3       OPCODE
3372 E96FE 1000          CON(5)  #001
           0
3373 E9703 1             CON(1)  1       GROUP C
3374 E9704 0             CON(1)  0
3375               *
3376               *
3377 E9705 50            CON(2)  5       FIXED
3378 E9707 5             CON(1)  5       OPCODE MNEMONIC
3379 E9708 05D3          NIBASC  \P=P+1 \
           05B2
           1302
3380 E9714 50            CON(2)  (F.LSET)!(F.TREE)
3381 E9716 2             CON(1)  2       OPCODE
3382 E9717 C000          CON(5)  #0C
           0
3383 E971C 00            CON(2)  0       NO VARIABLES
3384               *
3385               *
3386 E971E 10            CON(2)  1       REGISTER TESTS
3387 E9720 4             CON(1)  4       OPCODE MNEMONIC
3388 E9721 F324          NIBASC  \?B<C  \
           C334
           0202
3389 E972D 70            CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
3390 E972F 3             CON(1)  3       OPCODE
3391 E9730 5080          CON(5)  #805
           0
3392 E9735 2             CON(1)  2       GROUP B
3393 E9736 0             CON(1)  0
3394               *
3395               *
3396 E9737 20            CON(2)  2       REGISTER ARITHMETIC
3397 E9739 4             CON(1)  4       OPCODE MNEMONIC
```

```
3398 E973A 3414          NIBASC  \CAEX  \
           5485
           0202
3399 E9746 40            CON(2)  (F.TREE)
3400 E9748 3             CON(1)  3       OPCODE
3401 E9749 E000          CON(5)  #00E
           0
3402 E974E 2             CON(1)  2       GROUP D
3403 E974F 0             CON(1)  0
```



```
3404                     STITLE  HASH TO GROUP 41
3405 E9750       OPCG41
3406 E9750 20            CON(2)  2       REGISTER ARITHMETIC
3407 E9752 4             CON(1)  4       OPCODE MNEMONIC
3408 E9753 44D3          NIBASC  \D=-D  \
           D244
           0202
3409 E975F 00            CON(2)  0
3410 E9761 3             CON(1)  3       OPCODE
3411 E9762 B000          CON(5)  #00B
           0
3412 E9767 4             CON(1)  4       GROUP F
3413 E9768 0             CON(1)  0
3414             *
3415             *
3416 E9769 20            CON(2)  2       REGISTER ARITHMETIC
3417 E976B 5             CON(1)  5       OPCODE MNEMONIC
3418 E976C 34D3          NIBASC  \C=C-B \
           34D2
           2402
3419 E9778 00            CON(2)  0
3420 E977A 3             CON(1)  3       OPCODE
3421 E977B 9000          CON(5)  #009
           0
3422 E9780 3             CON(1)  3       GROUP E
3423 E9781 0             CON(1)  0
3424             *
3425             *
3426 E9782 50            CON(2)  5       FIXED
3427 E9784 6             CON(1)  6       OPCODE MNEMONIC
3428 E9785 2535          NIBASC  \RSTK=C\
           45B4
           D334
3429 E9791 10            CON(2)  (F.LSET)
3430 E9793 2             CON(1)  2       OPCODE
3431 E9794 6000          CON(5)  #06
           0
3432 E9799 00            CON(2)  0       NO VARIABLES
3433             *
3434             *
3435 E979B 30            CON(2)  3       REGISTER LOGIC
3436 E979D 5             CON(1)  5       OPCODE MNEMONIC
3437 E979E 24D3          NIBASC  \B=C!B \
           3412
           2402
3438 E97AA 10            CON(2)  (F.LSET)
3439 E97AC 4             CON(1)  4       OPCODE
3440 E97AD 90E0          CON(5)  #0E09
           0
3441 E97B2 00            CON(2)  0       NO VARIABLES
3442             *
3443             *
3444 E97B4 40            CON(2)  4       BRANCHES
3445 E97B6 6             CON(1)  6       OPCODE MNEMONIC
3446 E97B7 34F4          NIBASC  \CON(3)\
```

```
            E482
            3392
3447 E97C3 53          CON(2)  (F.LSET)!(F.TREE)!(F.ABSL)!(F.EXPR)
3448 E97C5 3           CON(1)  3       OPCODE
3449 E97C6 0000        CON(5)  #0
            0
3450 E97CB 1           CON(1)  1       FILL.MARK
3451 E97CC 3           CON(1)  3       FILL.LENGTH
3452            *
3453            *
3454 E97CD 40          CON(2)  4       BRANCHES
3455 E97CF 6           CON(1)  6       OPCODE MNEMONIC
3456 E97D0 2554        NIBASC  \REL(5)\
            C482
            5392
3457 E97DC 52          CON(2)  (F.LSET)!(F.TREE)!(F.EXPR)
3458 E97DE 5           CON(1)  5       OPCODE
3459 E97DF 0000        CON(5)  #0
            0
3460 E97E4 1           CON(1)  1       FILL.MARK
3461 E97E5 5           CON(1)  5       FILL.LENGTH
3462            *
3463            *
3464 E97E6 81          CON(2)  24      TITLE PSEUDO-OP
3465 E97E8 5           CON(1)  5       OPCODE MNEMONIC
3466 E97E9 4594        NIBASC  \TITLE \
            45C4
            5402
3467 E97F5 50          CON(2)  (F.LSET)!(F.TREE)
3468 E97F7 0           CON(1)  0       OPCODE
3469 E97F8 0000        CON(5)  #0
            0
3470 E97FD 00          CON(2)  0       NO VARIABLES
3471            *
3472            *
3473 E97FF 10          CON(2)  1       REGISTER TESTS
3474 E9801 4           CON(1)  4       OPCODE MNEMONIC
3475 E9802 F324        NIBASC  \?B<A  \
            C314
            0202
3476 E980E 70          CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
3477 E9810 3           CON(1)  3       OPCODE
3478 E9811 0080        CON(5)  #800
            0
3479 E9816 2           CON(1)  2       GROUP B
3480 E9817 0           CON(1)  0
3481            *
3482            *
3483 E9818 50          CON(2)  5       FIXEDB
3484 E981A 4           CON(1)  4       OPCODE MNEMONIC
3485 E981B 2435        NIBASC  \BSLC  \
            C434
            0202
3486 E9827 50          CON(2)  (F.LSET)!(F.TREE)
3487 E9829 3           CON(1)  3       OPCODE
```



```
3488 E982A 1180        CON(5)  #811
            0
3489 E982F 00          CON(2)  0       NO VARIABLES
```

```
3490                    STITLE  HASH TO GROUP 42
3491 E9831      OPCG42
3492 E9831 50           CON(2)  5       FIXED
3493 E9833 6            CON(1)  6       OPCODE MNEMONIC
3494 E9834 3554         NIBASC  \SETHEX\
           4584
           5485
3495 E9840 10           CON(2)  (F.LSET)
3496 E9842 2            CON(1)  2       OPCODE
3497 E9843 4000         CON(5)  #04
           0
3498 E9848 00           CON(2)  0       NO VARIABLES
3499            *
3500            *
3501 E984A 20           CON(2)  2       REGISTER ARITHMETIC
3502 E984C 5            CON(1)  5       OPCODE MNEMONIC
3503 E984D 34D3         NIBASC  \C=C+B \
           34B2
           2402
3504 E9859 00           CON(2)  0
3505 E985B 3            CON(1)  3       OPCODE
3506 E985C 9000         CON(5)  #009
           0
3507 E9861 1            CON(1)  1       GROUP C
3508 E9862 0            CON(1)  0
3509            *
3510            *
3511 E9863 50           CON(2)  5       FIXED
3512 E9865 5            CON(1)  5       OPCODE MNEMONIC
3513 E9866 3525         NIBASC  \SREQ? \
           5415
           F302
3514 E9872 50           CON(2)  (F.LSET)!(F.TREE)
3515 E9874 3            CON(1)  3       OPCODE
3516 E9875 E080         CON(5)  #80E
           0
3517 E987A 00           CON(2)  0       NO VARIABLES
3518            *
3519            *
3520 E987C 10           CON(2)  1       REGISTER TESTS
3521 E987E 5            CON(1)  5       OPCODE MNEMONIC
3522 E987F F344         NIBASC  \?D<=C \
           C3D3
           3402
3523 E988B 70           CON(2)  (F.LSET)!(F.RETY)!(F.TREE)
3524 E988D 3            CON(1)  3       OPCODE
3525 E988E F080         CON(5)  #80F
           0
3526 E9893 2            CON(1)  2       GROUP B
3527 E9894 0            CON(1)  0
3528            *
3529            *
3530 E9895 50           CON(2)  5       FIXED
3531 E9897 5            CON(1)  5       OPCODE MNEMONIC
3532 E9898 3435         NIBASC  \CSTEX \
```



```
           4554
           8502
3533 E98A4 50           CON(2)  (F.LSET)!(F.TREE)
3534 E98A6 2            CON(1)  2       OPCODE
3535 E98A7 B000         CON(5)  #0B
           0
3536 E98AC 00           CON(2)  0       NO VARIABLES
```

```
3537                    STITLE  FIT STATS
3538            zSIZE   EQU     (*)-zTHIS
3539            zLEFT   EQU     (zNEXT)-*
3540 E98AE              BSS     zLEFT
3541 E9900              END
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER


```
F.ABSL  Abs      16 #00010 -   17  268  465  570  609  675 1398 1541
                             1873 2117 2202 2268 2305 2438 2691 2836
                             2904 3360 3447
F.EXPR  Abs      32 #00020 -   18   89  153  172  192  201  259  268
                              372  381  400  429  465  570  609  675
                              767  776  852  871  880  899 1002 1020
                             1077 1152 1180 1199 1312 1369 1398 1436
                             1455 1541 1561 1570 1664 1712 1778 1825
                             1843 1873 1987 2071 2108 2117 2127 2164
                             2202 2268 2305 2381 2390 2438 2467 2505
                             2663 2672 2691 2796 2836 2866 2885 2904
                             3094 3113 3169 3226 3360 3447 3457
F.GOYS  Abs       8 #00008 -   16  172  353
F.GSUB  Abs      64 #00040 -   19 2390 2672
F.LSET  Abs       1 #00001 -   13   71   80   89   98  108  117  126
                              135  144  162  172  182  192  201  211
                              220  230  240  250  268  278  287  316
                              325  344  353  362  372  409  447  456
                              465  475  485  494  503  512  522  551
                              570  590  599  609  619  628  637  647
                              656  675  685  694  703  712  721  730
                              758  767  776  786  796  815  824  843
                              852  871  880  889  908  917  926  945
                              973  983  992 1011 1020 1030 1039 1048
                             1077 1086 1115 1124 1134 1143 1161 1180
                             1189 1218 1227 1237 1246 1264 1273 1292
                             1321 1330 1398 1427 1436 1445 1455 1474
                             1483 1493 1503 1541 1561 1570 1579 1598
                             1607 1627 1664 1674 1712 1731 1740 1759
                             1778 1788 1798 1834 1853 1873 1883 1893
                             1902 1941 1950 1959 1968 1978 1996 2005
                             2014 2034 2044 2053 2062 2071 2080 2089
                             2098 2117 2136 2145 2155 2164 2174 2183
                             2193 2202 2222 2231 2240 2249 2259 2268
                             2287 2296 2305 2315 2334 2343 2352 2362
                             2371 2381 2390 2400 2409 2428 2438 2448
                             2458 2467 2505 2514 2524 2533 2542 2551
                             2560 2579 2588 2597 2616 2626 2635 2663
                             2672 2682 2691 2701 2710 2719 2738 2757
                             2766 2796 2806 2816 2836 2866 2875 2904
                             2914 2933 2952 2961 2970 2979 2988 2997
                             3007 3045 3064 3094 3104 3113 3122 3131
                             3151 3189 3208 3217 3235 3274 3283 3302
                             3322 3332 3341 3351 3360 3380 3389 3429
                             3438 3447 3457 3467 3476 3486 3495 3514
                             3523 3533
F.PART  Abs     128 #00080 -   20  465  570  609 1541 1873 2117 2268
                             2305 2438 2691 3360
F.RETY  Abs       2 #00002 -   14   98  162  182  220  230  325  362
                              372  409  475  512  599  637  767  786
                              852  889  973  992 1124 1189 1227 1273
                             1292 1330 1427 1445 1455 1483 1493 1503
                             1570 1607 1788 1853 1883 1902 1968 2014
                             2034 2098 2145 2183 2249 2352 2428 2448
                             2514 2616 2719 2766 2806 2816 2875 2914
```

```
                                2997  3045  3064  3113  3131  3235  3302  3341
                                3389  3476  3523
F.TREE  Abs        4 #00004 -    15   108   117   126   135   144   220   230
                               240   250   259   268   278   287   334   344
                               353   362   372   409   419   429   438   503
                               512   522   531   541   551   560   619   628
                               637   647   656   665   712   721   730   739
                               749   786   796   805   815   889   899   908
                               917   926   935   945   954   992  1002  1011
                              1020  1057  1067  1077  1086  1124  1134  1143
                              1152  1208  1218  1227  1237  1246  1255  1302
                              1312  1321  1330  1340  1378  1388  1398  1408
                              1474  1483  1493  1503  1512  1522  1532  1588
                              1598  1607  1617  1627  1636  1674  1683  1693
                              1703  1731  1740  1749  1788  1798  1807  1816
                              1843  1853  1863  1902  1912  1922  1932  1968
                              1978  1987  1996  2034  2044  2053  2062  2089
                              2098  2108  2164  2174  2183  2193  2202  2212
                              2249  2259  2268  2278  2324  2334  2343  2352
                              2362  2390  2400  2409  2448  2458  2467  2476
                              2514  2524  2533  2542  2597  2606  2616  2626
                              2635  2644  2691  2701  2710  2719  2729  2786
                              2796  2806  2816  2826  2836  2885  2894  2904
                              2914  2924  2979  2988  2997  3007  3016  3026
                              3064  3074  3084  3094  3131  3141  3151  3160
                              3217  3226  3235  3245  3255  3265  3312  3322
                              3332  3341  3351  3380  3389  3399  3447  3457
                              3467  3476  3486  3514  3523  3533
*NULENT  Abs   947590 #E7586 -   132
 OPCG00  Abs   947415 #E74D7 -    67    23
 OPCG01  Abs   947640 #E75B8 -   149    24
 OPCG02  Abs   948015 #E772F -   292    25
 OPCG03  Abs   948240 #E7810 -   377    26
 OPCG04  Abs   948415 #E78BF -   443    27
 OPCG05  Abs   948740 #E7A04 -   566    28
 OPCG06  Abs   949015 #E7B17 -   671    29
 OPCG07  Abs   949240 #E7BF8 -   754    30
 OPCG08  Abs   949415 #E7CA7 -   820    31
 OPCG09  Abs   949790 #E7E1E -   959    32
 OPCG10  Abs   949965 #E7ECD -  1026    33
 OPCG11  Abs   950140 #E7F7C -  1091    34
 OPCG12  Abs   950315 #E802B -  1157    35
 OPCG13  Abs   950590 #E813E -  1260    36
 OPCG14  Abs   950815 #E821F -  1345    37
 OPCG15  Abs   950990 #E82CE -  1413    38
 OPCG16  Abs   951315 #E8413 -  1537    39
 OPCG17  Abs   951590 #E8526 -  1641    40
 OPCG18  Abs   951765 #E85D5 -  1708    41
 OPCG19  Abs   951890 #E8652 -  1755    42
 OPCG20  Abs   952065 #E8701 -  1821    43
 OPCG21  Abs   952190 #E877E -  1869    44
 OPCG22  Abs   952365 #E882D -  1937    45
 OPCG23  Abs   952540 #E88DC -  2001    46
 OPCG24  Abs   952715 #E898B -  2067    47
 OPCG25  Abs   952840 #E8A08 -  2113    48
```



```
 OPCG26  Abs   953115 #E8B1B -  2218    49
 OPCG27  Abs   953290 #E8BCA -  2283    50
 OPCG28  Abs   953515 #E8CAB -  2367    51
 OPCG29  Abs   953640 #E8D28 -  2414    52
 OPCG30  Abs   953815 #E8DD7 -  2481    53
 OPCG31  Abs   953990 #E8E86 -  2547    54
 OPCG32  Abs   954265 #E8F99 -  2649    55
 OPCG33  Abs   954490 #E907A -  2734    56
 OPCG34  Abs   954765 #E918D -  2842    57
 OPCG35  Abs   954990 #E926E -  2929    58
 OPCG36  Abs   955265 #E9381 -  3031    59
 OPCG37  Abs   955440 #E9430 -  3100    60
 OPCG38  Abs   955615 #E94DF -  3165    61
 OPCG39  Abs   955890 #E95F2 -  3270    62
 OPCG40  Abs   956115 #E96D3 -  3356    63
 OPCG41  Abs   956240 #E9750 -  3405    64
 OPCG42  Abs   956465 #E9831 -  3491    65
=OPTABL  Abs   947200 #E7400 -    23
 zLEFT   Abs       82 #00052 -  3539  3540
 zNEXT   Abs   956672 #E9900 -     4  3539
 zSIZE   Abs     9390 #024AE -  3538
 zTHIS   Abs   947200 #E7400 -     3  3538
```



```
Input Parameters

  Source file name is MR&AS0

  Listing file name is MR/AS0::65

  Object file name is MR%AS0::65

                                        111111
                              0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News
```

```
 1                      TITLE  AS1:_FORTH_ASSEMBLER_VARS
 2                      RDSYMB MR%GTO
 3 E9900                ABS    #E9900
 4              zTHIS   EQU    *
 5              zNEXT   EQU    #EA300
 6              ********************::*********************************
 7              ***
 8              *** MR&AS1     <840504.1037>
 9              ***
10              ***
11              ***  FORTH ASSEMBLER
12              ***  VARIABLES, SOME LOW LEVEL ROUTINES
13              ***
14              ******************************************************
15              ************************************************************
16              ***
17              ***  +BASE
18              ***
19              *** ( NUM -- PTR )
20              *** ( CONVERT AN OFFSET INTO THE VARIABLE BUFFER TO )
21              *** ( AN ACTUAL ADDRESS )
22              ************************************************************
23 E9900 0000 =a+BASE CON(5) =DOCOL          : +BASE
          0
24              *
25              * TRY TO FIND VARIABLE BUFFER
26              *
27 E9905 0000         CON(5) =VARID          VARID
          0
28 E990A 0000         CON(5) =AT             @
          0
29 E990F 0000         CON(5) =IOFIND         FINDBF
          0
30 E9914 0000         CON(5) =?DUP           ?DUP
          0
31 E9919 0000         CON(5) =ZBRNH          IF
          0
32 E991E F000         REL(5) %IF1
          0
33              *
34              * FOUND.  ADD BUFFER ADDRESS TO OFFSET
35              *
36 E9923 0000         CON(5) =ADD            +
          0
37 E9928 0000         CON(5) =SEMI           ;
          0
38              *
39              * NOT FOUND.  ABORT WITH Assembler Error.
40              * THIS SHOULD NEVER HAPPEN.  IT WAS PUT IN DURING
41              * TESTING BUT NEVER TAKEN OUT.  IF IT DID OCCUR,
42              * THEN SOMEHOW THE VARIABLE BUFFER GOT TRASHED
43              * OR SOMETHING WAS BEING DONE WITH A VARIABLE
44              * AFTER IT WAS KILLED.  -Geoff
45              *
46 E992D        %IF1
```



```
47              *
48              * ABORT WITH MESSAGE
49              *
50 E992D 0000         CON(5) =CR             CR
          0
51 E9932 0000         CON(5) =PDOTQ          ." Assembler Error"
          0
52 E9937 F0           CON(2) 15
53 E9939 1437         NIBASC \Assemble\
          3756
          D626
          C656
54 E9949 2702         NIBASC \r Error\
          5427
          27F6
          27
55 E9957 0000         CON(5) =aCLEAN         CLEANUP
          0
56 E995C 0000         CON(5) =ABORT          ABORT
          0
57              ************************************************************
58              ***
59              ***  +STR
60              ***
61              *** ( NUM -- STR )
62              *** ( CONVERT AN OFFSET INTO THE VARIABLE BUFFER TO )
63              *** ( A STRING ADDRESS & COUNT )
64              ************************************************************
65 E9961 0000 =a+STR  CON(5) =DOCOL          : +STR
          0
66 E9966 0000         CON(5) =TWO+           2+
          0
67 E996B 0099         CON(5) =a+BASE         +BASE
          E
68 E9970 0000         CON(5) =COUNT          COUNT
          0
69 E9975 0000         CON(5) =SEMI           ;          ( * STR * )
          0
70              ************************************************************
71              ***
72              ***  PASS
73              ***
74              ************************************************************
75 E997A 0000 =aPASS  CON(5) =DOCOL          : PASS
          0
76 E997F 0000         CON(5) =ZERO           0
          0
77 E9984 0099         CON(5) =a+BASE         +BASE
          E
78 E9989 0000         CON(5) =SEMI           ;
          0
79              ************************************************************
80              ***
81              ***  LC
82              ***
```

```
                ****************************************************************
                ***
                ***  LAST.REQ
                ***
                ****************************************************************
E99A7 0000 =aLSTRQ CON(5) =DOCOL         : LAST.REQ
      0
E99AC 0000         CON(5) =LIT           10
      0
E99B1 A000         CON(5) 10
      0
E99B6 0099         CON(5) =a+BASE        +BASE
      E
E99BB 0000         CON(5) =SEMI          ;
      0
                ****************************************************************
                ***
                ***  OCCUR
                ***
                *** ( 1=LISTING FILE FULL MESSAGE )
                *** ( 2=FILE TYPE DECLARED )
                *** ( 4=SYMBOL TABLE FULL )
                *** ( 8=ERROR OCCURED THIS SOURCE LINE )
                *** (16=TOKEN KEYWORD ENCOUNTERED )
                ****************************************************************
E99C0 0000 =aOCCUR CON(5) =DOCOL         : OCCUR
      0
E99C5 0000         CON(5) =LIT           15
      0
E99CA F000         CON(5) 15
      0
E99CF 0099         CON(5) =a+BASE        +BASE
      E
E99D4 0000         CON(5) =SEMI          ;
      0
                ****************************************************************
                ***
                ***  KNOWN
                ***
                ****************************************************************
E99D9 0000 =aKNOWN CON(5) =DOCOL         : KNOWN
      0
E99DE 0000         CON(5) =LIT           20
      0
```



```
E99E3 4100         CON(5) 20
      0
E99E8 0099         CON(5) =a+BASE        +BASE
      E
E99ED 0000         CON(5) =SEMI          ;
      0
                ****************************************************************
                ***
                ***  DONE
                ***
                ****************************************************************
E99F2 0000 =aDONE  CON(5) =DOCOL         : DONE
      0
E99F7 0000         CON(5) =LIT           25
      0
E99FC 9100         CON(5) 25
      0
E9A01 0099         CON(5) =a+BASE        +BASE
      E
E9A06 0000         CON(5) =SEMI          ;
      0
                ****************************************************************
                ***
                ***  FILETYPE
                ***
                ****************************************************************
E9A0B 0000 =aFLTYP CON(5) =DOCOL         : FILETYPE
      0
E9A10 0000         CON(5) =LIT           30
      0
E9A15 E100         CON(5) 30
      0
E9A1A 0099         CON(5) =a+BASE        +BASE
      E
E9A1F 0000         CON(5) =SEMI          ;
      0
                ****************************************************************
                ***
                ***  LINECNT
                ***
                ****************************************************************
E9A24 0000 =aLINEC CON(5) =DOCOL         : LINECNT
      0
E9A29 0000         CON(5) =LIT           280
      0
E9A2E 8110         CON(5) 280
      0
E9A33 0099         CON(5) =a+BASE        +BASE
      E
E9A38 0000         CON(5) =SEMI          ;
      0
                ****************************************************************
                ***
                ***  ERROR COUNT
                ***
```

```
158           ****************************************************************
159 E9A3D 0000 =aERCNT CON(5) =DOCOL         : ERROR COUNT
        0
160 E9A42 0000         CON(5) =LIT           285
        0
161 E9A47 D110         CON(5) 285
        0
162 E9A4C 0099         CON(5) =a+BASE        +BASE
        E
163 E9A51 0000         CON(5) =SEMI          ;
        0
164           ****************************************************************
165           ***
166           ***  TIT
167           ***
168           ****************************************************************
169 E9A56 0000 =aTIT   CON(5) =DOCOL         : TIT
        0
170 E9A5B 0000         CON(5) =LIT           290
        0
171 E9A60 2210         CON(5) 290
        0
172 E9A65 1699         CON(5) =a+STR         +STR
        E
173 E9A6A 0000         CON(5) =SEMI          ;
        0
174           ****************************************************************
175           ***
176           ***  STIT
177           ***
178           ****************************************************************
179 E9A6F 0000 =aSTIT  CON(5) =DOCOL         : STIT
        0
180 E9A74 0000         CON(5) =LIT           374
        0
181 E9A79 6710         CON(5) 374
        0
182 E9A7E 1699         CON(5) =a+STR         +STR
        E
183 E9A83 0000         CON(5) =SEMI          ;
        0
184           ****************************************************************
185           ***
186           ***  SLINE#
187           ***
188           ****************************************************************
189 E9A88 0000 =aSLIN# CON(5) =DOCOL         : SLINE#
        0
190 E9A8D 0000         CON(5) =LIT           35
        0
191 E9A92 3200         CON(5) 35
        0
192 E9A97 0099         CON(5) =a+BASE        +BASE
        E
193 E9A9C 0000         CON(5) =SEMI          ;
```



```
        0
194           ****************************************************************
195           ***
196           ***  SFILE
197           ***
198           ****************************************************************
199 E9AA1 0000 =aSFILE CON(5) =DOCOL         : SFILE
        0
200 E9AA6 0000         CON(5) =LIT           458
        0
201 E9AAB AC10         CON(5) 458
        0
202 E9AB0 1699         CON(5) =a+STR         +STR
        E
203 E9AB5 0000         CON(5) =SEMI          ;
        0
204           ****************************************************************
205           ***
206           ***  SLINE
207           ***
208           ****************************************************************
209 E9ABA 0000 =aSLINE CON(5) =DOCOL         : SLINE
        0
210 E9ABF 0000         CON(5) =LIT           502
        0
211 E9AC4 6F10         CON(5) 502
        0
212 E9AC9 1699         CON(5) =a+STR         +STR
        E
213 E9ACE 0000         CON(5) =SEMI          ;
        0
214           ****************************************************************
215           ***
216           ***  LNOFF
217           ***
218           ****************************************************************
219 E9AD3 0000 =aLNOFF CON(5) =DOCOL         : LNOFF
        0
220 E9AD8 0000         CON(5) =LIT           40
        0
221 E9ADD 8200         CON(5) 40
        0
222 E9AE2 0099         CON(5) =a+BASE        +BASE
        E
223 E9AE7 0000         CON(5) =SEMI          ;
        0
224           ****************************************************************
225           ***
226           ***  LLINE#
227           ***
228           ****************************************************************
229 E9AEC 0000 =aLLIN# CON(5) =DOCOL         : LLINE#
        0
230 E9AF1 0000         CON(5) =LIT           45
        0
```

```
231 E9AF6 D200          CON(5) 45
          0
232 E9AFB 0099          CON(5) =a+BASE          +BASE
          E
233 E9B00 0000          CON(5) =SEMI            ;
          0
234             ****************************************************************
235             ***
236             ***  LLINE
237             ***
238             ****************************************************************
239 E9B05 0000 =aLLINE CON(5) =DOCOL          : LLINE
          0
240 E9B0A 0000          CON(5) =LIT             698
          0
241 E9B0F AB20          CON(5) 698
          0
242 E9B14 1699          CON(5) =a+STR           +STR
          E
243 E9B19 0000          CON(5) =SEMI            ;
          0
244             ****************************************************************
245             ***
246             ***  LISTADDR
247             ***
248             ****************************************************************
249 E9B1E 0000 =aLSTAD CON(5) =DOCOL          : LISTADDR
          0
250 E9B23 0000          CON(5) =LIT             50
          0
251 E9B28 2300          CON(5) 50
          0
252 E9B2D 0099          CON(5) =a+BASE          +BASE
          E
253 E9B32 0000          CON(5) =SEMI            ;
          0
254             ****************************************************************
255             ***
256             ***  LISTFILE
257             ***
258             ****************************************************************
259 E9B37 0000 =aLISTF CON(5) =DOCOL          : LISTFILE
          0
260 E9B3C 0000          CON(5) =LIT             55
          0
261 E9B41 7300          CON(5) 55
          0
262 E9B46 0099          CON(5) =a+BASE          +BASE
          E
263 E9B4B 0000          CON(5) =SEMI            ;
          0
264             ****************************************************************
265             ***
266             ***  PAGENO
267             ***
```



```
268             ****************************************************************
269 E9B50 0000 =aPAGE# CON(5) =DOCOL          : PAGENO
          0
270 E9B55 0000          CON(5) =LIT             60
          0
271 E9B5A C300          CON(5) 60
          0
272 E9B5F 0099          CON(5) =a+BASE          +BASE
          E
273 E9B64 0000          CON(5) =SEMI            ;
          0
274             ****************************************************************
275             ***
276             ***  TOLIST
277             ***
278             ****************************************************************
279 E9B69 0000 =aTOLST CON(5) =DOCOL          : TOLIST
          0
280 E9B6E 0000          CON(5) =LIT             65
          0
281 E9B73 1400          CON(5) 65
          0
282 E9B78 0099          CON(5) =a+BASE          +BASE
          E
283 E9B7D 0000          CON(5) =SEMI            ;
          0
284             ****************************************************************
285             ***
286             ***  OFILE
287             ***
288             ****************************************************************
289 E9B82 0000 =aOFILE CON(5) =DOCOL          : OFILE
          0
290 E9B87 0000          CON(5) =LIT             894
          0
291 E9B8C E730          CON(5) 894
          0
292 E9B91 1699          CON(5) =a+STR           +STR
          E
293 E9B96 0000          CON(5) =SEMI            ;
          0
294             ****************************************************************
295             ***
296             ***  ERR$
297             ***
298             ****************************************************************
299 E9B9B 0000 =aERR$  CON(5) =DOCOL          : ERR$
          0
300 E9BA0 0000          CON(5) =LIT             938
          0
301 E9BA5 AA30          CON(5) 938
          0
302 E9BAA 1699          CON(5) =a+STR           +STR
          E
303 E9BAF 0000          CON(5) =SEMI            ;
```

```
          0
  304           ****************************************************************
  305           ***
  306           ***  LABEL.FIELD
  307           ***
  308           ****************************************************************
  309 E9BB4 0000 =aLAB.F CON(5) =DOCOL         : LABEL.FIELD
          0
  310 E9BB9 0000         CON(5) =LIT           1042
          0
  311 E9BBE 2140         CON(5) 1042
          0
  312 E9BC3 1699         CON(5) =a+STR         +STR
          E
  313 E9BC8 0000         CON(5) =SEMI          ;
          0
  314           ****************************************************************
  315           ***
  316           ***  OPCODE.FIELD
  317           ***
  318           ****************************************************************
  319 E9BCD 0000 =aOPC.F CON(5) =DOCOL         : OPCODE.FIELD
          0
  320 E9BD2 0000         CON(5) =LIT           1066
          0
  321 E9BD7 A240         CON(5) 1066
          0
  322 E9BDC 1699         CON(5) =a+STR         +STR
          E
  323 E9BE1 0000         CON(5) =SEMI          ;
          0
  324           ****************************************************************
  325           ***
  326           ***  EXPRESSION.FIELD
  327           ***
  328           ****************************************************************
  329 E9BE6 0000 =aEXP.F CON(5) =DOCOL         : EXPRESSION.FIELD
          0
  330 E9BEB 0000         CON(5) =LIT           1090
          0
  331 E9BF0 2440         CON(5) 1090
          0
  332 E9BF5 1699         CON(5) =a+STR         +STR
          E
  333 E9BFA 0000         CON(5) =SEMI          ;
          0
  334           ****************************************************************
  335           ***
  336           ***  EXPRCOL
  337           ***
  338           ****************************************************************
  339 E9BFF 0000 =aEXPRC CON(5) =DOCOL         : EXPRCOL
          0
  340 E9C04 0000         CON(5) =LIT           215
          0
```



```
  341 E9C09 7D00         CON(5) 215
          0
  342 E9C0E 0099         CON(5) =a+BASE        +BASE
          E
  343 E9C13 0000         CON(5) =SEMI          ;
          0
  344           *** ( *** THESE VARIABLES ARE SET WHEN AN OPCODE LOOKUP IS )
  345           *** (     MADE, BY THE WORD 'OPC.LOOKUP' *** )
  346           ****************************************************************
  347           ***
  348           ***  OPTYPE
  349           ***
  350           ****************************************************************
  351 E9C18 0000 =aOPTYP CON(5) =DOCOL         : OPTYPE
          0
  352 E9C1D 0000         CON(5) =LIT           70
          0
  353 E9C22 6400         CON(5) 70
          0
  354 E9C27 0099         CON(5) =a+BASE        +BASE
          E
  355 E9C2C 0000         CON(5) =SEMI          ;
          0
  356           ****************************************************************
  357           ***
  358           ***  OPFLAGS
  359           ***
  360           ****************************************************************
  361 E9C31 0000 =aOPFLG CON(5) =DOCOL         : OPFLAGS
          0
  362 E9C36 0000         CON(5) =LIT           75
          0
  363 E9C3B B400         CON(5) 75
          0
  364 E9C40 0099         CON(5) =a+BASE        +BASE
          E
  365 E9C45 0000         CON(5) =SEMI          ;
          0
  366           ****************************************************************
  367           ***
  368           ***  OPLEN
  369           ***
  370           ****************************************************************
  371 E9C4A 0000 =aOPLEN CON(5) =DOCOL         : OPLEN
          0
  372 E9C4F 0000         CON(5) =LIT           80
          0
  373 E9C54 0500         CON(5) 80
          0
  374 E9C59 0099         CON(5) =a+BASE        +BASE
          E
  375 E9C5E 0000         CON(5) =SEMI          ;
          0
  376           ****************************************************************
  377           ***
```

```
                ***  OP
                ***
                ********************************************************************
E9C63 0000 =aOP     CON(5) =DOCOL          : OP
           0
E9C68 0000          CON(5) =LIT            85
           0
E9C6D 5500          CON(5) 85
           0
E9C72 0099          CON(5) =a+BASE         +BASE
           E
E9C77 0000          CON(5) =SEMI           ;
           0
                ********************************************************************
                ***
                ***  FILL.MARK
                ***
                ********************************************************************
E9C7C 0000 =aFILMK CON(5) =DOCOL           : FILL.MARK
           0
E9C81 0000          CON(5) =LIT            175
           0
E9C86 FA00          CON(5) 175
           0
E9C8B 0099          CON(5) =a+BASE         +BASE
           E
E9C90 0000          CON(5) =SEMI           ;
           0
                ********************************************************************
                ***
                ***  REG.GROUP
                ***
                ********************************************************************
E9C95 0000 =aREGRP CON(5) =DOCOL           : REG.GROUP
           0
E9C9A 0000          CON(5) =LIT            175
           0
E9C9F FA00          CON(5) 175
           0
E9CA4 0099          CON(5) =a+BASE         +BASE
           E
E9CA9 0000          CON(5) =SEMI           ;
           0
                ********************************************************************
                ***
                ***  ARITH.GROUP
                ***
                ********************************************************************
E9CAE 0000 =aARGRP CON(5) =DOCOL           : ARITH.GROUP
           0
E9CB3 0000          CON(5) =LIT            175
           0
E9CB8 FA00          CON(5) 175
           0
E9CBD 0099          CON(5) =a+BASE         +BASE
```



```
           E
E9CC2 0000          CON(5) =SEMI           ;
           0
                ********************************************************************
                ***
                ***  FILL.LENGTH
                ***
                ********************************************************************
E9CC7 0000 =aFILEN CON(5) =DOCOL           : FILL.LENGTH
           0
E9CCC 0000          CON(5) =LIT            180
           0
E9CD1 4B00          CON(5) 180
           0
E9CD6 0099          CON(5) =a+BASE         +BASE
           E
E9CDB 0000          CON(5) =SEMI           ;
           0
                ********************************************************************
                ***
                ***  OPCPOS & RTNSTK
                ***
                ********************************************************************
E9CE0               =aRSTK
E9CE0 0000 =aOPCPO CON(5) =DOCOL           : OPCPOS
           0
E9CE5 0000          CON(5) =LIT            185
           0
E9CEA 9B00          CON(5) 185
           0
E9CEF 0099          CON(5) =a+BASE         +BASE
           E
E9CF4 0000          CON(5) =SEMI           ;
           0
                ********************************************************************
                ***
                ***  OPCPTR & DATASTK
                ***
                ********************************************************************
E9CF9               =aDSTK
E9CF9 0000 =aOPCPT CON(5) =DOCOL           : OPCPTR
           0
E9CFE 0000          CON(5) =LIT            190
           0
E9D03 EB00          CON(5) 190
           0
E9D08 0099          CON(5) =a+BASE         +BASE
           E
E9D0D 0000          CON(5) =SEMI           ;
           0
                ********************************************************************
                ***
                ***  OPCTOP
                ***
                ********************************************************************
```

```
453 E9D12 0000 =aOPCTO CON(5) =DOCOL         : OPCTOP
          0
454 E9D17 0000         CON(5) =LIT           195
          0
455 E9D1C 3C00         CON(5) 195
          0
456 E9D21 0099         CON(5) =a+BASE        +BASE
          E
457 E9D26 0000         CON(5) =SEMI          ;
          0
458            ****************************************************************
459            ***
460            ***  VALUE
461            ***
462            ****************************************************************
463 E9D2B 0000 =aVALUE CON(5) =DOCOL         : VALUE
          0
464 E9D30 0000         CON(5) =LIT           200
          0
465 E9D35 8C00         CON(5) 200
          0
466 E9D3A 0099         CON(5) =a+BASE        +BASE
          E
467 E9D3F 0000         CON(5) =SEMI          ;
          0
468            ****************************************************************
469            ***
470            ***  TOKEN
471            ***
472            ****************************************************************
473 E9D44 0000 =aTOKEN CON(5) =DOCOL         : TOKEN
          0
474 E9D49 0000         CON(5) =LIT           205
          0
475 E9D4E DC00         CON(5) 205
          0
476 E9D53 0099         CON(5) =a+BASE        +BASE
          E
477 E9D58 0000         CON(5) =SEMI          ;
          0
478            ****************************************************************
479            ***
480            ***  IDENTIFIER
481            ***
482            ****************************************************************
483 E9D5D 0000 =aIDENT CON(5) =DOCOL         : IDENTIFIER
          0
484 E9D62 0000         CON(5) =LIT           1194
          0
485 E9D67 AA40         CON(5) 1194
          0
486 E9D6C 1699         CON(5) =a+STR         +STR
          E
487 E9D71 0000         CON(5) =SEMI          ;
          0
```



```
488            *** ( *** THESE ARE USED BY VARIOUS PSEUDO-OPS )
489            ****************************************************************
490            ***
491            ***  REVERS
492            ***
493            ****************************************************************
494 E9D76 0000 =aREVER CON(5) =DOCOL         : REVERS
          0
495 E9D7B 0000         CON(5) =LIT           210
          0
496 E9D80 2D00         CON(5) 210
          0
497 E9D85 0099         CON(5) =a+BASE        +BASE
          E
498 E9D8A 0000         CON(5) =SEMI          ;
          0
499            ****************************************************************
500            ***
501            ***  OLDEXPR
502            ***
503            ****************************************************************
504 E9D8F 0000 =aOLDEX CON(5) =DOCOL         : OLDEXPR
          0
505 E9D94 0000         CON(5) =LIT           1218
          0
506 E9D99 2C40         CON(5) 1218
          0
507 E9D9E 1699         CON(5) =a+STR         +STR
          E
508 E9DA3 0000         CON(5) =SEMI          ;
          0
509            ****************************************************************
510            ***
511            ***  HITOK
512            ***
513            ****************************************************************
514 E9DA8 0000 =aHITOK CON(5) =DOCOL         : HITOK
          0
515 E9DAD 0000         CON(5) =LIT           220
          0
516 E9DB2 CD00         CON(5) 220
          0
517 E9DB7 0099         CON(5) =a+BASE        +BASE
          E
518 E9DBC 0000         CON(5) =SEMI          ;
          0
519            ****************************************************************
520            ***
521            ***  LOTOK
522            ***
523            ****************************************************************
524 E9DC1 0000 =aLOTOK CON(5) =DOCOL         : LOTOK
          0
525 E9DC6 0000         CON(5) =LIT           225
          0
```

```
526 E9DCB 1E00          CON(5) 225
          0
527 E9DD0 0099          CON(5) =a+BASE          +BASE
          E
528 E9DD5 0000          CON(5) =SEMI            ;
          0
529             ****************************************************************
530             ***
531             ***  TENT
532             ***
533             ****************************************************************
534 E9DDA 0000 =aTENT   CON(5) =DOCOL           : TENT
          0
535 E9DDF 0000          CON(5) =LIT             230
          0
536 E9DE4 6E00          CON(5) 230
          0
537 E9DE9 0099          CON(5) =a+BASE          +BASE
          E
538 E9DEE 0000          CON(5) =SEMI            ;
          0
539             ****************************************************************
540             ***
541             ***  KENT
542             ***
543             ****************************************************************
544 E9DF3 0000 =aKENT   CON(5) =DOCOL           : KENT
          0
545 E9DF8 0000          CON(5) =LIT             235
          0
546 E9DFD BE00          CON(5) 235
          0
547 E9E02 0099          CON(5) =a+BASE          +BASE
          E
548 E9E07 0000          CON(5) =SEMI            ;
          0
549             ****************************************************************
550             ***
551             ***  SYTAD
552             ***
553             ****************************************************************
554 E9E0C 0000 =aSYTAD  CON(5) =DOCOL           : SYTAD
          0
555 E9E11 0000          CON(5) =LIT             240
          0
556 E9E16 0F00          CON(5) 240
          0
557 E9E1B 0099          CON(5) =a+BASE          +BASE
          E
558 E9E20 0000          CON(5) =SEMI            ;
          0
559             ****************************************************************
560             ***
561             ***  SYTSIZE
562             ***
```



```
563             ****************************************************************
564 E9E25 0000 =aSYTSZ CON(5) =DOCOL           : SYTSIZE
          0
565 E9E2A 0000          CON(5) =LIT             245
          0
566 E9E2F 5F00          CON(5) 245
          0
567 E9E34 0099          CON(5) =a+BASE          +BASE
          E
568 E9E39 0000          CON(5) =SEMI            ;
          0
569             ****************************************************************
570             ***
571             ***  SYTLAST
572             ***
573             ****************************************************************
574 E9E3E 0000 =aSYTLS  CON(5) =DOCOL           : SYTLAST
          0
575 E9E43 0000          CON(5) =LIT             250
          0
576 E9E48 AF00          CON(5) 250
          0
577 E9E4D 0099          CON(5) =a+BASE          +BASE
          E
578 E9E52 0000          CON(5) =SEMI            ;
          0
579             ****************************************************************
580             ***
581             ***  BTOP
582             ***
583             ****************************************************************
584 E9E57 0000 =aBTOP   CON(5) =DOCOL           : BTOP
          0
585 E9E5C 0000          CON(5) =LIT             255
          0
586 E9E61 FF00          CON(5) 255
          0
587 E9E66 0099          CON(5) =a+BASE          +BASE
          E
588 E9E6B 0000          CON(5) =SEMI            ;
          0
589             ****************************************************************
590             ***
591             ***  BLOW
592             ***
593             ****************************************************************
594 E9E70 0000 =aBLOW   CON(5) =DOCOL           : BLOW
          0
595 E9E75 0000          CON(5) =LIT             260
          0
596 E9E7A 4010          CON(5) 260
          0
597 E9E7F 0099          CON(5) =a+BASE          +BASE
          E
598 E9E84 0000          CON(5) =SEMI            ;
```

```
          0
  599           ****************************************************************
  600           ***
  601           ***  BFOUND
  602           ***
  603           ****************************************************************
  604 E9E89 0000 =aBFOUN CON(5) =DOCOL         : BFOUND
          0
  605 E9E8E 0000         CON(5) =LIT           265
          0
  606 E9E93 9010         CON(5) 265
          0
  607 E9E98 0099         CON(5) =a+BASE        +BASE
          E
  608 E9E9D 0000         CON(5) =SEMI          ;
          0
  609           ****************************************************************
  610           ***
  611           ***  OBJFILE
  612           ***
  613           ****************************************************************
  614 E9EA2 0000 =aOBJFL CON(5) =DOCOL         : OBJFILE
          0
  615 E9EA7 0000         CON(5) =LIT           270
          0
  616 E9EAC E010         CON(5) 270
          0
  617 E9EB1 0099         CON(5) =a+BASE        +BASE
          E
  618 E9EB6 0000         CON(5) =SEMI          ;
          0
  619           ****************************************************************
  620           ***
  621           ***  FILESIZE
  622           ***
  623           ****************************************************************
  624 E9EBB 0000 =aFLSIZ CON(5) =DOCOL         : FILESIZE
          0
  625 E9EC0 0000         CON(5) =LIT           275
          0
  626 E9EC5 3110         CON(5) 275
          0
  627 E9ECA 0099         CON(5) =a+BASE        +BASE
          E
  628 E9ECF 0000         CON(5) =SEMI          ;
          0
  629           ****************************************************************
  630           ***
  631           ***  ERRMSG
  632           ***
  633           *** ( NUM -- STR )
  634           *** ( GET A MESSAGE & ALLOW FOR FOREIGN LANGUAGE )
  635           *** ( POLL INTERCEPTS )
  636           ****************************************************************
  637 E9ED4 0000 =aERMSG CON(5) =DOCOL         : ERRMSG
```



```
          0
  638           *
  639           * GENERATE A STRING TO PASS TO BASIC WHICH HAS FORM:
  640           *      MSG$(47000+xx)
  641           * WHERE xx IS THE MESSAGE NUMBER PASSED TO ERRMSG.
  642           *
  643 E9ED9 0000         CON(5) =QUOTC         " MSG$(47000+"
          0
  644 E9EDE B0           CON(2) 11
  645 E9EE0 B0           CON(2) 11
  646 E9EE2 D435         NIBASC \MSG$(470\
          7442
          8243
          7303
  647 E9EF2 0303         NIBASC \00+\
          B2
  648 E9EF8 0000         CON(5) =a>PAD         >PAD          ( * NUM STR * )
          0
  649 E9EFD 0000         CON(5) =ROT           ROT           ( * STR NUM * )
          0
  650 E9F02 0000         CON(5) =DEC           DECIMAL
          0
  651 E9F07 0000         CON(5) =aNUM#2        #2
          0
  652 E9F0C 0000         CON(5) =S<&           S<&           ( * STR * )
          0
  653 E9F11 0000         CON(5) =QUOTC         " )"
          0
  654 E9F16 10           CON(2) 1
  655 E9F18 10           CON(2) 1
  656 E9F1A 92           NIBASC \)\
  657 E9F1C 0000         CON(5) =S<&           S<&           ( * STR * )
          0
  658           *
  659           * CALL BASIC FUNCTION MSG(), WHICH RESIDES IN THE EDITOR
  660           * ROM THAT RETURNS THE MESSAGE STRING FROM THE APPROPRIATE
  661           * MESSAGE TABLE (DIFFERS FROM ERRM$ IN THAT IT ISSUES A
  662           * MESSAGE POLL)
  663           *
  664 E9F21 AB2A         CON(5) =aBAS$         BASIC$        ( * STR * )
          E
  665 E9F26 0000         CON(5) =SEMI          ;             ( * STR * )
          0
  666           ****************************************************************
  667           ***
  668           ***  STR.INIT
  669           ***
  670           *** ( -- )
  671           *** ( INITIALIZE THE STRINGS IN THE VARIABLE BUFFER )
  672           ****************************************************************
  673 E9F2B 0000 =aSTRIN CON(5) =DOCOL         : STR.INIT
          0
  674           *
  675           * LABEL FIELD
  676           *
```

```
677 E9F30 0000          CON(5) =ZERO          0
          0
678 E9F35 0099          CON(5) =a+BASE        +BASE          ( * PTR * )
          E
679 E9F3A 0000          CON(5) =LIT           1042
          0
680 E9F3F 2140          CON(5) 1042
          0
681 E9F44 0000          CON(5) =OVER          OVER
          0
682 E9F49 0000          CON(5) =ADD           +
          0
683 E9F4E 0000          CON(5) =LIT           10
          0
684 E9F53 A000          CON(5) 10
          0
685 E9F58 0000          CON(5) =SWAP          SWAP
          0
686 E9F5D 0000          CON(5) =STORE         !              ( * PTR * )
          0
687             *
688             * OPCODE FIELD
689             *
690 E9F62 0000          CON(5) =LIT           1066
          0
691 E9F67 A240          CON(5) 1066
          0
692 E9F6C 0000          CON(5) =OVER          OVER
          0
693 E9F71 0000          CON(5) =ADD           +
          0
694 E9F76 0000          CON(5) =LIT           10
          0
695 E9F7B A000          CON(5) 10
          0
696 E9F80 0000          CON(5) =SWAP          SWAP
          0
697 E9F85 0000          CON(5) =STORE         !              ( * PTR * )
          0
698             *
699             * IDENTIFIER
700             *
701 E9F8A 0000          CON(5) =LIT           1194
          0
702 E9F8F AA40          CON(5) 1194
          0
703 E9F94 0000          CON(5) =OVER          OVER
          0
704 E9F99 0000          CON(5) =ADD           +
          0
705 E9F9E 0000          CON(5) =LIT           10
          0
706 E9FA3 A000          CON(5) 10
          0
707 E9FA8 0000          CON(5) =SWAP          SWAP
```



```
          0
708 E9FAD 0000          CON(5) =STORE         !              ( * PTR * )
          0
709             *
710             * SOURCE FILE NAME
711             *
712 E9FB2 0000          CON(5) =LIT           458
          0
713 E9FB7 AC10          CON(5) 458
          0
714 E9FBC 0000          CON(5) =OVER          OVER
          0
715 E9FC1 0000          CON(5) =ADD           +
          0
716 E9FC6 0000          CON(5) =LIT           20
          0
717 E9FCB 4100          CON(5) 20
          0
718 E9FD0 0000          CON(5) =SWAP          SWAP
          0
719 E9FD5 0000          CON(5) =STORE         !              ( * PTR * )
          0
720             *
721             * OBJECT FILE NAME
722             *
723 E9FDA 0000          CON(5) =LIT           894
          0
724 E9FDF E730          CON(5) 894
          0
725 E9FE4 0000          CON(5) =OVER          OVER
          0
726 E9FE9 0000          CON(5) =ADD           +
          0
727 E9FEE 0000          CON(5) =LIT           20
          0
728 E9FF3 4100          CON(5) 20
          0
729 E9FF8 0000          CON(5) =SWAP          SWAP
          0
730 E9FFD 0000          CON(5) =STORE         !              ( * PTR * )
          0
731             *
732             * TITLE STRING
733             *
734 EA002 0000          CON(5) =LIT           290
          0
735 EA007 2210          CON(5) 290
          0
736 EA00C 0000          CON(5) =OVER          OVER
          0
737 EA011 0000          CON(5) =ADD           +
          0
738 EA016 0000          CON(5) =LIT           40
          0
739 EA01B 8200          CON(5) 40
```

```
          0
740 EA020 0000          CON(5) =SWAP            SWAP
          0
741 EA025 0000          CON(5) =STORE           !               ( * PTR * )
          0
742               *
743               * SUB-TITLE STRING
744               *
745 EA02A 0000          CON(5) =LIT             374
          0
746 EA02F 6710          CON(5) 374
          0
747 EA034 0000          CON(5) =OVER            OVER
          0
748 EA039 0000          CON(5) =ADD             +
          0
749 EA03E 0000          CON(5) =LIT             40
          0
750 EA043 8200          CON(5) 40
          0
751 EA048 0000          CON(5) =SWAP            SWAP
          0
752 EA04D 0000          CON(5) =STORE           !               ( * PTR * )
          0
753               *
754               * ERROR STRING
755               *
756 EA052 0000          CON(5) =LIT             938
          0
757 EA057 AA30          CON(5) 938
          0
758 EA05C 0000          CON(5) =OVER            OVER
          0
759 EA061 0000          CON(5) =ADD             +
          0
760 EA066 0000          CON(5) =LIT             50
          0
761 EA06B 2300          CON(5) 50
          0
762 EA070 0000          CON(5) =SWAP            SWAP
          0
763 EA075 0000          CON(5) =STORE           !               ( * PTR * )
          0
764               *
765               * EXPRESSION FIELD
766               *
767 EA07A 0000          CON(5) =LIT             1090
          0
768 EA07F 2440          CON(5) 1090
          0
769 EA084 0000          CON(5) =OVER            OVER
          0
770 EA089 0000          CON(5) =ADD             +
          0
771 EA08E 0000          CON(5) =LIT             50
```



```
          0
772 EA093 2300          CON(5) 50
          0
773 EA098 0000          CON(5) =SWAP            SWAP
          0
774 EA09D 0000          CON(5) =STORE           !               ( * PTR * )
          0
775               *
776               * OLD EXPRESSION
777               *
778 EA0A2 0000          CON(5) =LIT             1218
          0
779 EA0A7 2C40          CON(5) 1218
          0
780 EA0AC 0000          CON(5) =OVER            OVER
          0
781 EA0B1 0000          CON(5) =ADD             +
          0
782 EA0B6 0000          CON(5) =LIT             50
          0
783 EA0BB 2300          CON(5) 50
          0
784 EA0C0 0000          CON(5) =SWAP            SWAP
          0
785 EA0C5 0000          CON(5) =STORE           !               ( * PTR * )
          0
786               *
787               * SOURCE LINE
788               *
789 EA0CA 0000          CON(5) =LIT             502
          0
790 EA0CF 6F10          CON(5) 502
          0
791 EA0D4 0000          CON(5) =OVER            OVER
          0
792 EA0D9 0000          CON(5) =ADD             +
          0
793 EA0DE 0000          CON(5) =LIT             96
          0
794 EA0E3 0600          CON(5) 96
          0
795 EA0E8 0000          CON(5) =SWAP            SWAP
          0
796 EA0ED 0000          CON(5) =STORE           !               ( * PTR * )
          0
797               *
798               * LISTING FILE LINE
799               *
800 EA0F2 0000          CON(5) =LIT             698
          0
801 EA0F7 AB20          CON(5) 698
          0
802 EA0FC 0000          CON(5) =ADD             +
          0
803 EA101 0000          CON(5) =LIT             96
```

```
          0
804 EA106 0600          CON(5) 96
          0
805 EA10B 0000          CON(5) =SWAP            SWAP
          0
806 EA110 0000          CON(5) =STORE           !
          0
807 EA115 0000          CON(5) =SEMI            ;               ( * * )
          0
808             ****************************************************************
809             ***
810             ***  VAR.INIT
811             ***
812             *** ( -- )
813             *** ( INITIALIZE THE VARIABLES WHICH THE ASSEMBLER USES )
814             ****************************************************************
815 EA11A 0000 =aVARIN CON(5) =DOCOL          : VAR.INIT
          0
816             *
817             * MAKE THE GENERAL PURPOSE BUFFER TO PUT VARIABLES IN
818             *
819 EA11F 0000          CON(5) =LIT             1332
          0
820 EA124 4350          CON(5) 1332
          0
821 EA129 0000          CON(5) =IOMAKE          MAKEBF
          0
822 EA12E 0000          CON(5) =ZBRNH           IF              ( * [ADDR ID] * )
          0
823 EA133 D200          REL(5) %IF2
          0
824             *
825             * SUCCESS!  SAVE THE BUFFER ID IN VARIABLE VARID
826             *
827 EA138 0000          CON(5) =VARID           VARID
          0
828 EA13D 0000          CON(5) =STORE           !               ( * ADDR * )
          0
829             *
830             * INITIALIZE THE NUMBERIC VARIABLES TO ZERO
831             *
832 EA142 0000          CON(5) =LIT             290
          0
833 EA147 2210          CON(5) 290
          0
834 EA14C 0000          CON(5) =ZERO            0
          0
835 EA151 0000          CON(5) =NFILL           NFILL           ( * * )
          0
836             *
837             * INITIALIZE THE STRINGS WITH MAX LENGTH & CURRENT LENGTH
838             *
839 EA156 B2F9          CON(5) =aSTRIN          STR.INIT        ( * * )
          E
840 EA15B 0000          CON(5) =SEMI            ;               ( * * )
```



```
          0
841             *
842             * COULDN'T BUILD THE BUFFER
843             *
844 EA160          %IF2                         ELSE            ( * * )
845 EA160 0000          CON(5) =LIT             47
          0
846 EA165 F200          CON(5) 47               'not enough room for assembler'
          0
847 EA16A 4DE9          CON(5) =aERMSG          ERRMSG
          E
848 EA16F 0000          CON(5) =TYPE            TYPE
          0
849 EA174 0000          CON(5) =ABORT           ABORT
          0
850             ****************************************************************
851             ***
852             ***  MEMMOVE
853             ***
854             *** ( -- )
855             *** ( ADJUST ALL POINTERS THAT ARE AFFECTED BY MEMORY )
856             *** ( MOVING )
857             ****************************************************************
858 EA179 0000 =aMEMOV CON(5) =DOCOL          : MEMMOVE
          0
859             *
860             * FIND OBJECT FILE START
861             *
862 EA17E 28B9          CON(5) =aOFILE          OFILE
          E
863 EA183 0000          CON(5) =SWAP            SWAP
          0
864 EA188 0000          CON(5) =DROP            DROP
          0
865 EA18D 0000          CON(5) =ZBRNH           IF
          0
866 EA192 9100          REL(5) %IF3
          0
867 EA197 28B9          CON(5) =aOFILE          OFILE
          E
868 EA19C 0000          CON(5) =FFIND           FFIND
          0
869 EA1A1 2AE9          CON(5) =aOBJFL          OBJFILE
          E
870 EA1A6 0000          CON(5) =STORE           !
          0
871 EA1AB          %IF3                         THEN
872             *
873             * FIND SYMBOL TABLE START
874             *
875 EA1AB 0000          CON(5) =QUOTC           " ASMBSY"
          0
876 EA1B0 60            CON(2) 6
877 EA1B2 60            CON(2) 6
878 EA1B4 1435          NIBASC \ASMBSY\
```

```
          D424
          3595
  879 EA1C0 0000        CON(5) =FFIND         FFIND
          0
  880 EA1C5 0000        CON(5) =LIT           37
          0
  881 EA1CA 5200        CON(5) 37
          0
  882 EA1CF 0000        CON(5) =ADD           +
          0
  883 EA1D4 C0E9        CON(5) =aSYTAD        SYTAD
          E
  884 EA1D9 0000        CON(5) =STORE         !
          0
  885           *
  886           * IS THERE A LISTING FILE ?
  887           *
  888 EA1DE 73B9        CON(5) =aLISTF        LISTFILE
          E
  889 EA1E3 0000        CON(5) =AT            @
          0
  890 EA1E8 0000        CON(5) =ZBRNH         IF
          0
  891 EA1ED 9100        REL(5) %IF4
          0
  892           *
  893           * FIND LISTING FILE START
  894           *
  895 EA1F2 0000        CON(5) =LISTNG        LISTING
          0
  896 EA1F7 0000        CON(5) =FFIND         FFIND
          0
  897 EA1FC 73B9        CON(5) =aLISTF        LISTFILE
          E
  898 EA201 0000        CON(5) =STORE         !
          0
  899 EA206      %IF4                         THEN
  900 EA206 0000        CON(5) =SEMI          ;             ( * * )
          0
  901           ***************************************************************
  902           ***
  903           ***  -1
  904           ***
  905           *** ( -- NUM )
  906           *** ( MINUS ONE IS USED SO OFTEN, IT'S WORTH IT )
  907           ***************************************************************
  908 EA20B 0000 =aNEG1  CON(5) =DOCOL        : -1
          0
  909 EA210 0000        CON(5) =LIT
          0
  910 EA215 FFFF        CON(5) 0-1            -1
          F
  911 EA21A 0000        CON(5) =SEMI          ;
          0
  912           ***************************************************************
```



```
  913           ***
  914           ***  ONTO PAD 2
  915           ***
  916           *** ( STR -- )
  917           *** ( MOVE THE STRING INTO THE SECOND HALF OF THE PAD )
  918           ***************************************************************
  919 EA21F 0000 =a>PAD2 CON(5) =DOCOL        : ONTO PAD 2
          0
  920           *
  921           * SAVE LENGTH
  922           *
  923 EA224 0000        CON(5) =>R            >R            ( * PTR * )
          0
  924           *
  925           * POINT TO START OF STRING (INCLUDE MAX & CUR LENGTH)
  926           *
  927 EA229 0000        CON(5) =LIT           4
          0
  928 EA22E 4000        CON(5) 4
          0
  929 EA233 0000        CON(5) =MINUS         -             ( * PTR * )
          0
  930           *
  931           * POINT TO SECOND HALF OF PAD
  932           *
  933 EA238 0000        CON(5) =PAD           PAD
          0
  934 EA23D 0000        CON(5) =LIT           80
          0
  935 EA242 0500        CON(5) 80
          0
  936 EA247 0000        CON(5) =ADD           +             ( * PTR PTR' * )
          0
  937           *
  938           * MOVE STRING
  939           *
  940 EA24C 0000        CON(5) =R>            R>
          0
  941 EA251 0000        CON(5) =TWO+          2+            ( * PTR PTR' CNT * )
          0
  942 EA256 0000        CON(5) =CMOVE         CMOVE
          0
  943 EA25B 0000        CON(5) =SEMI          ;             ( * * )
          0
  944           ***************************************************************
  945           ***
  946           ***  OFF PAD 2
  947           ***
  948           *** ( -- STR )
  949           *** ( RECOVER STRING FROM THE SECOND HALF OF THE PAD )
  950           ***************************************************************
  951 EA260 0000 =aPAD2> CON(5) =DOCOL        : OFF PAD 2
          0
  952           *
  953           * POINT TO SECOND HALF OF PAD
```

```
954                *
955 EA265 0000          CON(5) =PAD             PAD         ( * PTR * )
          0
956 EA26A 0000          CON(5) =LIT             82
          0
957 EA26F 2500          CON(5) 82
          0
958 EA274 0000          CON(5) =ADD             +           ( * PTR * )
          0
959                *
960                * EXTRACT STRING
961                *
962 EA279 0000          CON(5) =COUNT           COUNT       ( * STR * )
          0
963 EA27E 0000          CON(5) =SEMI            ;           ( * STR * )
          0
964                ****************************************************************
965                ***
966                ***  MEMORY ABORT
967                ***
968                *** ( -- )
969                *** ( ABORT ASSEMBLER BECAUSE MEMORY ALL GONE )
970                ****************************************************************
971 EA283 0000 =aMEMAB CON(5) =DOCOL          : MEMORY ABORT
          0
972                *
973                * PUT OBJECT FILE NAME INTO LATTER HALF OF PAD
974                *
975 EA288 28B9          CON(5) =aOFILE          OBJFILE     ( * STR * )
          E
976 EA28D F12A          CON(5) =a>PAD2          >PAD2       ( * * )
          E
977                *
978                * KILL OUR VARIABLE BUFFER SO WE HAVE SOME SPACE
979                *
980 EA292 0000          CON(5) =VARID           VARID
          0
981 EA297 0000          CON(5) =AT              @           ( * NUM * )
          0
982 EA29C 0000          CON(5) =IOKILL          KILLBF
          0
983 EA2A1 0000          CON(5) =DROP            DROP        ( * * )
          0
984                *
985                * ABORT WITH NOT ENOUGH MEMORY
986                *
987 EA2A6 0000          CON(5) =LIT             47
          0
988 EA2AB F200          CON(5) 47               'not enough memory for assembler'
          0
989 EA2B0 062A          CON(5) =aPAD2>          PAD2>       ( * NUM STR * )
          E
990 EA2B5 0000          CON(5) =aABRT           ASSEMBLER ABORT
          0
991                ****************************************************************
```



```
992                ***
993                ***  BASIC$
994                ***
995                *** ( STR -- STR )
996                *** ( DO A NORMAL 'BASIC$', BUT WITH ERROR TRAPS )
997                ****************************************************************
998 EA2BA 0000 =abAS$  CON(5) =DOCOL          : BASIC$
          0
999 EA2BF 0000          CON(5) =LIT             MEMORY ABORT
          0
1000 EA2C4 382A         CON(5) =aMEMAB
          E
1001 EA2C9 0000         CON(5) =ONERR           ONERR
          0
1002 EA2CE 0000         CON(5) =STORE           !
          0
1003 EA2D3 0000         CON(5) =BASIC$          BASIC$
          0
1004 EA2D8 0000         CON(5) =ZERO            0
          0
1005 EA2DD 0000         CON(5) =ONERR           ONERR
          0
1006 EA2E2 0000         CON(5) =STORE           !
          0
1007 EA2E7 0000         CON(5) =SEMI            ;           ( * STR * )
          0
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER

```
1008                    STITLE FIT STATS
1009            zSIZE   EQU    (*)-zTHIS
1010            zLEFT   EQU    (zNEXT)-*
1011 EA2EC              BSS    zLEFT
```



```
%IF1    Abs  956717 #E992D -    46    32
%IF2    Abs  958816 #EA160 -   844   823
%IF3    Abs  958891 #EA1AB -   871   866
%IF4    Abs  958982 #EA206 -   899   891
>R      Ext                -   923
?DUP    Ext                -    30
ABORT   Ext                -    56   849
ADD     Ext                -    36   682   693   704   715   726   737   748
                             759   770   781   792   802   882   936   958
AT      Ext                -    28   889   981
BASIC$  Ext                -  1003
CMOVE   Ext                -   942
COUNT   Ext                -    68   962
CR      Ext                -    50
DEC     Ext                -   650
DOCOL   Ext                -    23    65    75    84    94   109   119   129
                             139   149   159   169   179   189   199   209
                             219   229   239   249   259   269   279   289
                             299   309   319   329   339   351   361   371
                             381   391   401   411   421   432   443   453
                             463   473   483   494   504   514   524   534
                             544   554   564   574   584   594   604   614
                             624   637   673   815   858   908   919   951
                             971   998
DROP    Ext                -   864   983
FFIND   Ext                -   868   879   896
IOFIND  Ext                -    29
IOKILL  Ext                -   982
IOMAKE  Ext                -   821
LISTNG  Ext                -   895
LIT     Ext                -    85    95   110   120   130   140   150   160
                             170   180   190   200   210   220   230   240
                             250   260   270   280   290   300   310   320
                             330   340   352   362   372   382   392   402
                             412   422   433   444   454   464   474   484
                             495   505   515   525   535   545   555   565
                             575   585   595   605   615   625   679   683
                             690   694   701   705   712   716   723   727
                             734   738   745   749   756   760   767   771
                             778   782   789   793   800   803   819   832
                             845   880   909   927   934   956   987   999
MINUS   Ext                -   929
NFILL   Ext                -   835
ONERR   Ext                -  1001  1005
OVER    Ext                -   681   692   703   714   725   736   747   758
                             769   780   791
PAD     Ext                -   933   955
PDOTQ   Ext                -    51
QUOTC   Ext                -   643   653   875
R>      Ext                -   940
ROT     Ext                -   649
S<&     Ext                -   652   657
SEMI    Ext                -    37    69    78    88    98   113   123   133
                             143   153   163   173   183   193   203   213
                             223   233   243   253   263   273   283   293
```

```
                                303  313  323  333  343  355  365  375
                                385  395  405  415  425  436  447  457
                                467  477  487  498  508  518  528  538
                                548  558  568  578  588  598  608  618
                                628  665  807  840  900  911  943  963
                               1007
 STORE    Ext                 -  686  697  708  719  730  741  752  763
                                774  785  796  806  828  870  884  898
                               1002 1006
 SWAP     Ext                 -  685  696  707  718  729  740  751  762
                                773  784  795  805  863
 TWO+     Ext                 -   66  941
 TYPE     Ext                 -  848
 VARID    Ext                 -   27  827  980
 ZBRNH    Ext                 -   31  822  865  890
 ZERO     Ext                 -   76  677  834 1004
=a+BASE   Abs  956672 #E9900  -   23   67   77   87   97  112  122  132
                                142  152  162  192  222  232  252  262
                                272  282  342  354  364  374  384  394
                                404  414  424  435  446  456  466  476
                                497  517  527  537  547  557  567  577
                                587  597  607  617  627  678
=a+STR    Abs  956769 #E9961  -   65  172  182  202  212  242  292  302
                                312  322  332  486  507
 a>PAD    Ext                 -  648
=a>PAD2   Abs  959007 #EA21F  -  919  976
 aABRT    Ext                 -  990
=aARGRP   Abs  957614 #E9CAE  -  411
=aBAS$    Abs  959162 #EA2BA  -  998  664
=aBFOUN   Abs  958089 #E9E89  -  604
=aBLOW    Abs  958064 #E9E70  -  594
=aBTOP    Abs  958039 #E9E57  -  584
 aCLEAN   Ext                 -   55
=aDONE    Abs  956914 #E99F2  -  129
=aDSTK    Abs  957689 #E9CF9  -  442
=aERCNT   Abs  956989 #E9A3D  -  159
=aERMSG   Abs  958164 #E9ED4  -  637  847
=aERR$    Abs  957339 #E9B9B  -  299
=aEXP.F   Abs  957414 #E9BE6  -  329
=aEXPRC   Abs  957439 #E9BFF  -  339
=aFILEN   Abs  957639 #E9CC7  -  421
=aFILMK   Abs  957564 #E9C7C  -  391
=aFLSIZ   Abs  958139 #E9EBB  -  624
=aFLTYP   Abs  956939 #E9A0B  -  139
=aHITOK   Abs  957864 #E9DA8  -  514
=aIDENT   Abs  957789 #E9D5D  -  483
=aKENT    Abs  957939 #E9DF3  -  544
=aKNOWN   Abs  956889 #E99D9  -  119
=aLAB.F   Abs  957364 #E9BB4  -  309
=aLC      Abs  956814 #E998E  -   84
=aLINEC   Abs  956964 #E9A24  -  149
=aLISTF   Abs  957239 #E9B37  -  259  888  897
=aLLIN#   Abs  957164 #E9AEC  -  229
=aLLINE   Abs  957189 #E9B05  -  239
=aLNOFF   Abs  957139 #E9AD3  -  219
```



```
=aLOTOK   Abs  957889 #E9DC1  -  524
=aLSTAD   Abs  957214 #E9B1E  -  249
=aLSTRQ   Abs  956839 #E99A7  -   94
=aMEMAB   Abs  959107 #EA283  -  971 1000
=aMEMOV   Abs  958841 #EA179  -  858
=aNEG1    Abs  958987 #EA20B  -  908
 aNUM#2   Ext                 -  651
=aOBJFL   Abs  958114 #E9EA2  -  614  869
=aOCCUR   Abs  956864 #E99C0  -  109
=aOFILE   Abs  957314 #E9B82  -  289  862  867  975
=aOLDEX   Abs  957839 #E9D8F  -  504
=aOP      Abs  957539 #E9C63  -  381
=aOPC.F   Abs  957389 #E9BCD  -  319
=aOPCPO   Abs  957664 #E9CE0  -  432
=aOPCPT   Abs  957689 #E9CF9  -  443
=aOPCTO   Abs  957714 #E9D12  -  453
=aOPFLG   Abs  957489 #E9C31  -  361
=aOPLEN   Abs  957514 #E9C4A  -  371
=aOPTYP   Abs  957464 #E9C18  -  351
=aPAD2>   Abs  959072 #EA260  -  951  989
=aPAGE#   Abs  957264 #E9B50  -  269
=aPASS    Abs  956794 #E997A  -   75
=aREGRP   Abs  957589 #E9C95  -  401
=aREVER   Abs  957814 #E9D76  -  494
=aRSTK    Abs  957664 #E9CE0  -  431
=aSFILE   Abs  957089 #E9AA1  -  199
=aSLIN#   Abs  957064 #E9A88  -  189
=aSLINE   Abs  957114 #E9ABA  -  209
=aSTIT    Abs  957039 #E9A6F  -  179
=aSTRIN   Abs  958251 #E9F2B  -  673  839
=aSYTAD   Abs  957964 #E9E0C  -  554  883
=aSYTLS   Abs  958014 #E9E3E  -  574
=aSYTSZ   Abs  957989 #E9E25  -  564
=aTENT    Abs  957914 #E9DDA  -  534
=aTIT     Abs  957014 #E9A56  -  169
=aTOKEN   Abs  957764 #E9D44  -  473
=aTOLST   Abs  957289 #E9B69  -  279
=aVALUE   Abs  957739 #E9D2B  -  463
=aVARIN   Abs  958746 #EA11A  -  815
 zLEFT    Abs      20 #00014  - 1010 1011
 zNEXT    Abs  959232 #EA300  -    5 1010
 zSIZE    Abs    2540 #009EC  - 1009
 zTHIS    Abs  956672 #E9900  -    4 1009
```

nput Parameters

Source file name is GN&AS1

Listing file name is GN/AS1::65

Object file name is GN%AS1::65

```
                                  111111
                        0123456789012345
```

Initial flag settings are

rrors

None

aturn Assembler News

```
                        TITLE  AS2:_FORTH_ASSEMBLER_IO
                        RDSYMB MR%GTO
EA300                   ABS    #EA300
                zTHIS   EQU    *
                zNEXT   EQU    #EB200

                ********************************************************
                ***
                *** MR&AS2      <840504.1037>
                ***
                ***
                ***  FORTH ASSEMBLER
                ***  SOURCE, LISTING, OBJECT IO
                ***
                ********************************************************
                ****************************************************************
                ***
                ***  11BLANK
                ***
                *** ( -- STR )
                *** ( PUT ELEVEN SPACES IN A STRING )
                ****************************************************************
EA300 0000 =aBLNKB CON(5) =DOCOL         : 11BLANK
           0
EA305 0000         CON(5) =QUOTC         "           "
           0
EA30A B0           CON(2) 11
EA30C B0           CON(2) 11
EA30E 0202         NIBASC \        \
           0202
           0202
           0202
EA31E 0202         NIBASC \   \
           02
EA324 0000         CON(5) =SEMI          ;           ( * STR * )
           0
                ****************************************************************
                ***
                ***  5BLANK
                ***
                *** ( -- STR )
                *** ( PUT FIVE SPACES IN A STRING )
                ****************************************************************
EA329 0000 =aBLNK5 CON(5) =DOCOL         : 5BLANK
           0
EA32E 003A         CON(5) =aBLNKB        11BLANK
           E
EA333 0000         CON(5) =DROP          DROP
           0
EA338 0000         CON(5) =LIT           5
           0
EA33D 5000         CON(5) 5
           0
EA342 0000         CON(5) =SEMI          ;           ( * STR * )
           0
```



```
                ****************************************************************
                ***
                ***  2BLANK
                ***
                *** ( -- STR )
                *** ( PUT TWO SPACES IN A STRING )
                ****************************************************************
EA347 0000 =aBLNK2 CON(5) =DOCOL         : 2BLANK
           0
EA34C 923A         CON(5) =aBLNK5        5BLANK
           E
EA351 0000         CON(5) =DROP          DROP
           0
EA356 0000         CON(5) =TWO           2
           0
EA35B 0000         CON(5) =SEMI          ;           ( * STR * )
           0
                ****************************************************************
                ***
                ***  1BLANK
                ***
                *** ( -- STR )
                *** ( PUT ONE SPACE IN A STRING )
                ****************************************************************
EA360 0000 =aBLNK1 CON(5) =DOCOL         : 1BLANK
           0
EA365 743A         CON(5) =aBLNK2        2BLANK
           E
EA36A 0000         CON(5) =ONE-          1-
           0
EA36F 0000         CON(5) =SEMI          ;           ( * STR * )
           0
                ****************************************************************
                ***
                ***  4DUP
                ***
                *** ( STR STR' -- STR STR' STR STR' )
                *** ( DUPLICATE TWO STRINGS )
                ****************************************************************
EA374 0000 =aDUP4  CON(5) =DOCOL         : 4DUP
           0
EA379 0000         CON(5) =OVER2         2OVER
           0
EA37E 0000         CON(5) =OVER2         2OVER
           0
EA383 0000         CON(5) =SEMI          ;           ( * STR STR' STR STR' *
           0
                ****************************************************************
                ***
                ***  4DROP
                ***
                *** ( STR STR' -- )
                *** ( DROP TWO STRINGS )
                ****************************************************************
EA388 0000 =aDROP4 CON(5) =DOCOL         : 4DROP
```

```
          0
    85 EA38D 0000         CON(5) =2DROP         2DROP
          0
    86 EA392 0000         CON(5) =2DROP         2DROP
          0
    87 EA397 0000         CON(5) =SEMI          ;           ( * * )
          0
    88            ***************************************************************
    89            ***
    90            ***  >PAD
    91            ***
    92            *** ( STR -- STR )
    93            *** ( MOVE THE STRING TO THE PAD )
    94            ***************************************************************
    95 EA39C 0000 =a>PAD  CON(5) =DOCOL         : >PAD
          0
    96 EA3A1 0000         CON(5) =NULL$         NULL$
          0
    97 EA3A6 0000         CON(5) =S>&           S>&
          0
    98 EA3AB 0000         CON(5) =SEMI          ;           ( * STR * )
          0
    99            ***************************************************************
   100            ***
   101            ***  &CLEAR
   102            ***
   103            *** ( -- )
   104            *** ( CLEAR LISTING LINE )
   105            ***************************************************************
   106 EA3B0 0000 =a&CLR  CON(5) =DOCOL         : &CLEAR
          0
   107 EA3B5 0000         CON(5) =NULL$         NULL$
          0
   108 EA3BA 0000         CON(5) =aLLINE        LLINE
          0
   109 EA3BF 0000         CON(5) =S!            S!
          0
   110 EA3C4 0000         CON(5) =SEMI          ;           ( * * )
          0
   111           ***************************************************************
   112           ***
   113           ***  SET.FLAG
   114           ***
   115           *** ( NUM -- )
   116           *** ( SET FLAG IN VARIABLE OCCUR )
   117           ***************************************************************
   118 EA3C9 0000 =aSETF  CON(5) =DOCOL         : SET.FLAG
          0
   119 EA3CE 0000         CON(5) =aOCCUR        OCCUR       ( * NUM PTR * )
          0
   120 EA3D3 0000         CON(5) =AT            @           ( * NUM NUM' * )
          0
   121 EA3D8 0000         CON(5) =OR            OR          ( * NUM * )
          0
   122 EA3DD 0000         CON(5) =aOCCUR        OCCUR       ( * NUM PTR * )
```



```
          0
   123 EA3E2 0000         CON(5) =STORE         !           ( * * )
          0
   124 EA3E7 0000         CON(5) =SEMI          ;           ( * * )
          0
   125           ***************************************************************
   126           ***
   127           ***  TEST.FLAG
   128           ***
   129           *** ( NUM -- F )
   130           *** ( TEST FLAG IN VARIABLE OCCUR )
   131           ***************************************************************
   132 EA3EC 0000 =aTESTF CON(5) =DOCOL         : TEST.FLAG
          0
   133 EA3F1 0000         CON(5) =aOCCUR        OCCUR       ( * NUM PTR * )
          0
   134 EA3F6 0000         CON(5) =AT            @           ( * NUM NUM' * )
          0
   135 EA3FB 0000         CON(5) =AND           AND         ( * F * )
          0
   136 EA400 0000         CON(5) =SEMI          ;           ( * F * )
          0
   137           ***************************************************************
   138           ***
   139           ***  SP
   140           ***
   141           *** ( -- )
   142           *** ( PUT A SPACE IN A NUMBER FORMAT STRING )
   143           ***************************************************************
   144 EA405 0000 =aSP    CON(5) =DOCOL         : SP
          0
   145 EA40A 0000         CON(5) =LIT           32
          0
   146 EA40F 0200         CON(5) 32
          0
   147 EA414 0000         CON(5) =HOLD          HOLD
          0
   148 EA419 0000         CON(5) =SEMI          ;           ( * * )
          0
   149           ***************************************************************
   150           ***
   151           ***  #1
   152           ***
   153           *** ( NUM -- STR )
   154           *** ( FORMAT A NUMBER WITH ONE DIGIT )
   155           ***************************************************************
   156 EA41E 0000 =aNUM#1 CON(5) =DOCOL         : #1
          0
   157 EA423 0000         CON(5) =ZERO          0           ( * D.NUM * )
          0
   158 EA428 0000         CON(5) =LT#           <#
          0
   159 EA42D 0000         CON(5) =BRNCH         JUMP
          0
   160 EA432 2800         REL(5) %NUM1
```

```
                0
    161         ****************************************************************
    162         ***
    163         ***  #2
    164         ***
    165         *** ( NUM -- STR )
    166         *** ( FORMAT A NUMBER WITH TWO DIGITS )
    167         ****************************************************************
    168 EA437 0000 =aNUM#2 CON(5) =DOCOL         : #2
                0
    169 EA43C 0000         CON(5) =ZERO          0           ( * D.NUM * )
                0
    170 EA441 0000         CON(5) =LT#           <#
                0
    171 EA446 0000         CON(5) =BRNCH         JUMP
                0
    172 EA44B 4600         REL(5) %NUM2
                0
    173         ****************************************************************
    174         ***
    175         ***  #3
    176         ***
    177         *** ( NUM -- STR )
    178         *** ( FORMAT A NUMBER WITH THREE DIGITS )
    179         ****************************************************************
    180 EA450 0000 =aNUM#3 CON(5) =DOCOL         : #3
                0
    181 EA455 0000         CON(5) =ZERO          0           ( * D.NUM * )
                0
    182 EA45A 0000         CON(5) =LT#           <#
                0
    183 EA45F 504A         CON(5) =aSP           32 HOLD
                E
    184 EA464 0000         CON(5) =BRNCH         JUMP
                0
    185 EA469 1400         REL(5) %NUM3
                0
    186         ****************************************************************
    187         ***
    188         ***  #4
    189         ***
    190         *** ( NUM -- STR )
    191         *** ( FORMAT A NUMBER WITH FOUR DIGITS )
    192         ****************************************************************
    193 EA46E 0000 =aNUM#4 CON(5) =DOCOL         : #4
                0
    194 EA473 0000         CON(5) =ZERO          0           ( * D.NUM * )
                0
    195 EA478 0000         CON(5) =LT#           <#
                0
    196 EA47D 504A         CON(5) =aSP           32 HOLD
                E
    197 EA482 0000         CON(5) =BRNCH         JUMP
                0
    198 EA487 E100         REL(5) %NUM4
```



```
                0
    199         ****************************************************************
    200         ***
    201         ***  #5
    202         ***
    203         *** ( NUM -- STR )
    204         *** ( FORMAT A NUMBER WITH FIVE DIGITS )
    205         ****************************************************************
    206 EA48C 0000 =aNUM#5 CON(5) =DOCOL         : #5
                0
    207 EA491 0000         CON(5) =ZERO          0           ( * D.NUM * )
                0
    208 EA496 0000         CON(5) =LT#           <#
                0
    209 EA49B 504A         CON(5) =aSP           SP
                E
    210 EA4A0 0000         CON(5) =N#            #
                0
    211 EA4A5 0000  %NUM4  CON(5) =N#            #
                0
    212 EA4AA 0000  %NUM3  CON(5) =N#            #
                0
    213 EA4AF 0000  %NUM2  CON(5) =N#            #
                0
    214 EA4B4 0000  %NUM1  CON(5) =N#            #
                0
    215 EA4B9 0000         CON(5) =GT#           #>          ( * STR * )
                0
    216 EA4BE 0000         CON(5) =SEMI          ;           ( * STR * )
                0
    217         ****************************************************************
    218         ***
    219         ***  >OPx
    220         ***
    221         *** ( I -- ADDR )
    222         *** ( MOVE TO THE Ith OPCODE )
    223         ****************************************************************
    224 EA4C3 0000 =a>OPx  CON(5) =DOCOL         : >OPx
                0
    225 EA4C8 0000         CON(5) =ONE-          1-
                0
    226 EA4CD 0000         CON(5) =LIT           5
                0
    227 EA4D2 5000         CON(5) 5
                0
    228 EA4D7 0000         CON(5) =MULT          *
                0
    229 EA4DC 0000         CON(5) =aOP           OP
                0
    230 EA4E1 0000         CON(5) =ADD           +
                0
    231 EA4E6 0000         CON(5) =SEMI          ;           ( * ADDR * )
                0
    232         ****************************************************************
    233         ***
```

```
234                *** KILL
235                ***
236                *** ( STR -- )
237                *** ( PURGE FILE GIVEN IN 'STR' )
238                ****************************************************************
239 EA4EB 0000 =aKILL  CON(5) =DOCOL          : KILL
          0
240                *
241                * GENEREATE PURGE STRING IN SECOND PART OF PAD
242                *
243 EA4F0 0000         CON(5) =QUOTC          " PURGE "
          0
244 EA4F5 60           CON(2) 6
245 EA4F7 60           CON(2) 6
246 EA4F9 0555         NIBASC \PURGE \
          2574
          5402
247 EA505 C93A         CON(5) =a>PAD          >PAD          ( * STR STR' * )
          E
248 EA50A 0000         CON(5) =SWAP2          2SWAP
          0
249 EA50F 0000         CON(5) =S<&            S<&           ( * STR * )
          0
250                *
251                * SET UP FOR BASICX.  IF MEMORY RUNS OUT, TRAP THAT ERROR.
252                *
253 EA514 0000         CON(5) =LIT            MEMORY ABORT
          0
254 EA519 0000         CON(5) =aMEMAB
          0
255 EA51E 0000         CON(5) =ONERR          ONERR
          0
256 EA523 0000         CON(5) =STORE          !
          0
257 EA528 0000         CON(5) =BASICX         BASICX
          0
258                *
259                * RESET ERROR TRAP
260                *
261 EA52D 0000         CON(5) =ZERO           0
          0
262 EA532 0000         CON(5) =ONERR          ONERR
          0
263 EA537 0000         CON(5) =STORE          !
          0
264 EA53C 0000         CON(5) =SEMI           ;             ( * * )
          0
265                ****************************************************************
266                ***
267                *** PURGE
268                ***
269                *** ( STR NUM -- F )
270                *** ( PURGE FILE 'STR' ONLY IF TYPE IS 'NUM' )
271                *** ( RETURN TF ONLY IF FILE NO LONGER PRESENT )
272                ****************************************************************
```



```
273 EA541 0000 =aPURGE CON(5) =DOCOL          : PURGE
          0
274                *
275                * CHECK IF FILE EXISTS ALREADY
276                *
277 EA546 0000         CON(5) =>R             >R            ( * STR * )
          0
278 EA54B 0000         CON(5) =2DUP           2DUP
          0
279 EA550 0000         CON(5) =FFIND          FFIND
          0
280 EA555 0000         CON(5) =?DUP           ?DUP
          0
281 EA55A 0000         CON(5) =ZBRNH          IF            ( * STR [ADDR] * )
          0
282 EA55F B400         REL(5) %IF6
          0
283                *
284                * IT DOES, SO CHECK THE FILE TYPE AGAINST THE NUMBER PASSED
285                *
286 EA564 0000         CON(5) =LIT            16
          0
287 EA569 0100         CON(5) 16
          0
288 EA56E 0000         CON(5) =ADD            +
          0
289 EA573 0000         CON(5) =4N@            4N@
          0
290 EA578 0000         CON(5) =R>             R>
          0
291 EA57D 0000         CON(5) =EQUAL          =
          0
292 EA582 0000         CON(5) =ZBRNH          IF            ( * STR * )
          0
293 EA587 4100         REL(5) %IF7
          0
294                *
295                * WE HAVE A MATCH, CALL THE KILL ROUTINE
296                *
297 EA58C BE4A         CON(5) =aKILL          KILL          ( * * )
          E
298 EA591 0000         CON(5) =aNEG1          -1
          0
299 EA596 0000         CON(5) =SEMI           ;             ( * TF * )
          0
300                *
301                * NO MATCH, RETURN CANNOT PURGE FILE
302                *
303 EA59B       %IF7                          ELSE          ( * STR * )
304 EA59B 0000         CON(5) =2DROP          2DROP
          0
305 EA5A0 0000         CON(5) =ZERO           0
          0
306 EA5A5 0000         CON(5) =SEMI           ;             ( * FF * )
          0
```

```
                *
                * FILE WASN'T PRESENT, RETURN TRUE (IT'S GONE)
                *
EA5AA            %IF6                       ELSE        ( * STR * )
EA5AA 0000              CON(5) =R>            R>
        0
EA5AF 0000              CON(5) =DROP          DROP
        0
EA5B4 0000              CON(5) =2DROP         2DROP       ( * * )
        0
EA5B9 0000              CON(5) =aNEG1         -1
        0
EA5BE 0000              CON(5) =SEMI          ;           ( * TF * )
        0
                ****************************************************************
                ***
                ***  REMOVE
                ***
                *** ( STR -- F )
                *** ( PURGE FILE 'STR' ONLY IF TYPE IS TEXT )
                *** ( RETURN TF ONLY IF FILE NO LONGER PRESENT )
                ****************************************************************
EA5C3 0000 =aREMOV CON(5) =DOCOL         : REMOVE
        0
EA5C8 0000              CON(5) =ONE           1           ( * STR NUM * )
        0
EA5CD 145A              CON(5) =aPURGE        PURGE       ( * F * )
        E
EA5D2 0000              CON(5) =SEMI          ;           ( * F * )
        0
                ****************************************************************
                ***
                ***  CLOSEDOWN
                ***
                *** ( -- )
                *** ( CLEAR UP BUFFERS ETC )
                ****************************************************************
EA5D7 0000 =aCLOSE CON(5) =DOCOL         : CLOSEDOWN
        0
                *
                * CLOSE SOURCE FILE
                *
EA5DC 0000              CON(5) =SCFIB         SCRFIB
        0
EA5E1 0000              CON(5) =AT            @
        0
EA5E6 0000              CON(5) =CLOSF         CLOSEF
        0
                *
                * RESET OUTER SCRFIB, IN CASE IN LOADF
                *
EA5EB 0000              CON(5) =LIT           SOURCE.FIB#
        0
EA5F0 88CF              CON(5) =oGRAB
        2
```



```
EA5F5 0000              CON(5) =AT            @
        0
EA5FA 0000              CON(5) =SCFIB         SCRFIB
        0
EA5FF 0000              CON(5) =STORE         !
        0
                *
                * CLEAN UP VARIABLE BUFFER
                *
EA604 0000              CON(5) =VARID         VARID
        0
EA609 0000              CON(5) =AT            @
        0
EA60E 0000              CON(5) =IOKILL        KILLBF
        0
EA613 0000              CON(5) =DROP          DROP        ( * * )
        0
EA618 0000              CON(5) =ZERO          0
        0
EA61D 0000              CON(5) =VARID         VARID
        0
EA622 0000              CON(5) =STORE         !           ( * * )
        0
                *
                * RESET VARIABLE BASE TO ORIGINAL VALUE
                *
EA627 0000              CON(5) =CSP           CSP
        0
EA62C 0000              CON(5) =AT            @
        0
EA631 0000              CON(5) =BASE          BASE
        0
EA636 0000              CON(5) =STORE         !           ( * * )
        0
                *
                * RE-ENABLE ATTN KEY
                *
EA63B 0000              CON(5) =ZERO          0
        0
EA640 0000              CON(5) =LIT           193601
        0
EA645 144F              CON(5) 193601
        2
EA64A 0000              CON(5) =N!            N!
        0
                *
                * RE-ENABLE ERRORS
                *
EA64F 0000              CON(5) =ZERO          0
        0
EA654 0000              CON(5) =ONERR         ONERR
        0
EA659 0000              CON(5) =STORE         !
        0
EA65E 0000              CON(5) =SEMI          ;           ( * * )
```

```
          0
  381           ****************************************************************
  382           ***
  383           ***  CLEANUP
  384           ***
  385           *** ( -- )
  386           *** ( CLEAR UP RAM AND OTHER CONDITIONS CREATED )
  387           *** ( BY RUNNING THE ASSEMBLER )
  388           ****************************************************************
  389 EA663 0000 =aCLEAN CON(5) =DOCOL         : CLEANUP
          0
  390           *
  391           * REMOVE OBJECT FILE (USING INTERNAL NAME)
  392           *
  393 EA668 0000         CON(5) =QUOTC         " ASMBOB"
          0
  394 EA66D 60           CON(2) 6
  395 EA66F 60           CON(2) 6
  396 EA671 1435         NIBASC \ASMBOB\
          D424
          F424
  397 EA67D 3C5A         CON(5) =aREMOV        REMOVE
          E
  398 EA682 0000         CON(5) =DROP          DROP        ( * * )
          0
  399           *
  400           * REMOVE SYMBOL TABLE
  401           *
  402 EA687 0000         CON(5) =QUOTC         " ASMBSY"
          0
  403 EA68C 60           CON(2) 6
  404 EA68E 60           CON(2) 6
  405 EA690 1435         NIBASC \ASMBSY\
          D424
          3595
  406 EA69C 3C5A         CON(5) =aREMOV        REMOVE
          E
  407 EA6A1 0000         CON(5) =DROP          DROP        ( * * )
          0
  408           *
  409           * CLOSE DOWN THE REST
  410           *
  411 EA6A6 7D5A         CON(5) =aCLOSE        CLOSEDOWN
          E
  412 EA6AB 0000         CON(5) =SEMI          ;           ( * * )
          0
  413           ****************************************************************
  414           ***
  415           ***  ASSEMBLER ABORT
  416           ***
  417           *** ( NUM STR -- )
  418           *** ( PURGE THE FILE IN 'STR' THEN ABORT WITH ERROR )
  419           *** ( IN NUM )
  420           ****************************************************************
  421 EA6B0 0000 =aABRT  CON(5) =DOCOL         : ASSEMBLER ABORT
```



```
          0
  422           *
  423           * FIRST KILL THE FILE
  424           *
  425 EA6B5 0000         CON(5) =DUP           DUP
          0
  426 EA6BA 0000         CON(5) =ZBRNH         IF          ( * NUM STR * )
          0
  427 EA6BF F000         REL(5) %IF304
          0
  428 EA6C4 0000         CON(5) =2DUP          2DUP
          0
  429 EA6C9 BE4A         CON(5) =aKILL         KILL
          E
  430 EA6CE       %IF304                       THEN
  431 EA6CE 0000         CON(5) =2DROP         2DROP       ( * NUM * )
          0
  432           *
  433           * REPORT THE ERROR MESSAGE
  434           *
  435 EA6D3 0000         CON(5) =aERMSG        ERRMSG      ( * STR * )
          0
  436 EA6D8 0000         CON(5) =CR            CR          ( * STR * )
          0
  437 EA6DD 0000         CON(5) =TYPE          TYPE        ( * * )
          0
  438           *
  439           * REMOVE THE LISTING FILE IF IT EXISTS
  440           *
  441 EA6E2 0000         CON(5) =LISTNG        LISTING
          0
  442 EA6E7 0000         CON(5) =FSTX          FSYNTAX
          0
  443 EA6EC 0000         CON(5) =ZBRNH         IF          ( * * )
          0
  444 EA6F1 F000         REL(5) %IF8
          0
  445 EA6F6 0000         CON(5) =LISTNG        LISTING
          0
  446 EA6FB 3C5A         CON(5) =aREMOV        REMOVE      ( * F * )
          E
  447           *
  448           * CLEAN UP THE REST
  449           *
  450 EA700       %IF8                         THEN        ( * * )
  451 EA700 366A         CON(5) =aCLEAN        CLEANUP
          E
  452 EA705 0000         CON(5) =ABORT         ABORT
          0
  453           ****************************************************************
  454           ***
  455           ***  SHUTDOWN
  456           ***
  457           *** ( NUM -- )
  458           *** ( REPORT ABORT MESSAGE AND CLEANUP )
```

```
                ****************************************************************
EA70A 0000 =aSHUTD CON(5) =DOCOL        : SHUTDOWN
           0
EA70F 0000         CON(5) =aOFILE       OBJFILE     ( * NUM STR * )
           0
EA714 0B6A         CON(5) =aABRT        ASSEMBLER ABORT
           E
                ****************************************************************
                ***
                ***  LIST.ABORT
                ***
                *** ( -- )
                *** ( COULDN'T FIND LISTING DEVICE )
                ****************************************************************
EA719 0000 =aLSTAB CON(5) =DOCOL        : LIST.ABORT
           0
EA71E 0000         CON(5) =LIT          49
           0
EA723 1300         CON(5) 49            'invalid listing file'
           0
EA728 A07A         CON(5) =aSHUTD       SHUTDOWN
           E
                ****************************************************************
                ***
                ***  LIST.INIT
                ***
                *** ( -- )
                *** ( OPEN LISTING FILE OR DEVICE )
                ****************************************************************
EA72D 0000 =aLSTIN CON(5) =DOCOL        : LIST.INIT
           0
                *
                * ENSURE PAGESIZE VARIABLE IS VALID
                *
EA732 0000         CON(5) =PAGESZ       PAGESIZE
           0
EA737 0000         CON(5) =AT           @           ( * NUM * )
           0
EA73C 0000         CON(5) =LIT          7
           0
EA741 7000         CON(5) 7
           0
EA746 0000         CON(5) =GT           >           ( * F * )
           0
EA74B 0000         CON(5) =ZBRNH        IF
           0
EA750 0910         REL(5) %IF9
           0
                *
                * CLEAR LISTING VARIABLES:
                *  LISTADDR - HPIL ADDRESS FOR LIST DEVICE
                *  LISTFILE - ADDRESS IN RAM OF LISTING FILE
                *
EA755 0000         CON(5) =ZERO         0
           0
```



```
EA75A 0000         CON(5) =aLSTAD       LISTADDR
           0
EA75F 0000         CON(5) =STORE        !           ( * * )
           0
EA764 0000         CON(5) =ZERO         0
           0
EA769 0000         CON(5) =aLISTF       LISTFILE
           0
EA76E 0000         CON(5) =STORE        !           ( * * )
           0
                *
                * LISTING SPECIFIED?
                *
EA773 0000         CON(5) =LISTNG       LISTING     ( * STR * )
           0
EA778 0000         CON(5) =?DUP         ?DUP
           0
EA77D 0000         CON(5) =ZBRNH        IF          ( * ADDR [LEN] * )
           0
EA782 4510         REL(5) %IF10
           0
                *
                * LISTING SPECIFIED, IS IT IN RAM?
                *
EA787 0000         CON(5) =2DUP         2DUP
           0
EA78C 0000         CON(5) =FSTX         FSYNTAX
           0
EA791 0000         CON(5) =ZBRNH        IF          ( * STR * )
           0
EA796 3700         REL(5) %IF11
           0
                *
                * RAM FILE SPECIFIED, CREATE A RAM FILE
                *
EA79B 0000         CON(5) =LIT          4
           0
EA7A0 4000         CON(5) 4
           0
EA7A5 0000         CON(5) =FCREAT       FCREATE
           0
EA7AA 0000         CON(5) =?DUP         ?DUP
           0
EA7AF 0000         CON(5) =ZBRNH        IF          ( * [ADDR] * )
           0
EA7B4 3110         REL(5) %IF12
           0
                *
                * RAM FILE CREATED, SAVE ADDRESS IN LISTFILE
                *
EA7B9 0000         CON(5) =DUP          DUP
           0
EA7BE 0000         CON(5) =aLISTF       LISTFILE
           0
EA7C3 0000         CON(5) =STORE        !           ( * ADDR * )
```

```
          0
  532             *
  533             * RAM FILE CREATED, STORE LAST RECORD MARK FFFF
  534             *
  535 EA7C8 0000          CON(5) =aNEG1          -1
          0
  536 EA7CD 0000          CON(5) =SWAP           SWAP
          0
  537 EA7D2 0000          CON(5) =LIT            37
          0
  538 EA7D7 5200          CON(5) 37
          0
  539 EA7DC 0000          CON(5) =ADD            +            ( * NUM ADDR * )
          0
  540 EA7E1 0000          CON(5) =2DUP           2DUP
          0
  541 EA7E6 0000          CON(5) =C!             C!
          0
  542 EA7EB 0000          CON(5) =TWO+           2+
          0
  543 EA7F0 0000          CON(5) =C!             C!           ( * * )
          0
  544             *
  545             * RAM FILE CREATED, MARK OFFSET INTO CURRENT RECORD
  546             *
  547 EA7F5 0000          CON(5) =ZERO           0
          0
  548 EA7FA 0000          CON(5) =aLNOFF         LNOFF
          0
  549 EA7FF 0000          CON(5) =STORE          !            ( * * )
          0
  550 EA804 0000          CON(5) =SEMI           ;            ( * * )
          0
  551             *
  552             * HPIL DEVICE SPECIFIED, IS IT VALID?  TRAP ON ERROR.
  553             *
  554 EA809        %IF11
  555 EA809 0000          CON(5) =LIT            ['']
          0
  556 EA80E 917A          CON(5) =aLSTAB         LIST.ABORT   ( * STR PTR * )
          E
  557 EA813 0000          CON(5) =ONERR          ONERR
          0
  558 EA818 0000          CON(5) =STORE          !            ( * STR * )
          0
  559             *
  560             * BUILD DEVADDR('<LISTING>') STRING AND CALL BASICF
  561             *
  562 EA81D 0000          CON(5) =QUOTC          " DEVADDR('"
          0
  563 EA822 90            CON(2) 9
  564 EA824 90            CON(2) 9
  565 EA826 4454          NIBASC \DEVADDR(\
          6514
          4444
```



```
          2582
  566 EA836 72            NIBASC \'\
  567 EA838 C93A          CON(5) =a>PAD          >PAD         ( * STR STR' * )
          E
  568 EA83D 0000          CON(5) =SWAP2          2SWAP
          0
  569 EA842 0000          CON(5) =S<&            S<&          ( * STR * )
          0
  570 EA847 0000          CON(5) =QUOTC          " ')"
          0
  571 EA84C 20            CON(2) 2
  572 EA84E 20            CON(2) 2
  573 EA850 7292          NIBASC \')\
  574 EA854 0000          CON(5) =S<&            S<&          ( * STR * )
          0
  575 EA859 0000          CON(5) =BASF           BASICF       ( * * )
          0
  576             *
  577             * CHECK IF DEVICE IS ON PRIMARY LOOP.
  578             *
  579 EA85E 0000          CON(5) =IP             IP
          0
  580 EA863 0000          CON(5) =LASTX          LASTX
          0
  581 EA868 0000          CON(5) =XEY?           X=Y?
          0
  582 EA86D 0000          CON(5) =ZBRNH          IF
          0
  583 EA872 0500          REL(5) %IF13
          0
  584             *
  585             * DEVICE NOT ON A SECONDARY LOOP.  RECOVER FROM BASICF
  586             *
  587 EA877 0000          CON(5) =FTOI           FTOI         ( * NUM * )
          0
  588 EA87C 0000          CON(5) =FROP           FDROP
          0
  589 EA881 0000          CON(5) =FROP           FDROP
          0
  590 EA886 0000          CON(5) =ZERO           0
          0
  591 EA88B 0000          CON(5) =ONERR          ONERR
          0
  592 EA890 0000          CON(5) =STORE          !            ( * NUM * )
          0
  593             *
  594             * CHECK IF DEVICE FOUND.
  595             *
  596 EA895 0000          CON(5) =DUP            DUP
          0
  597 EA89A 0000          CON(5) =aNEG1          -1
          0
  598 EA89F 0000          CON(5) =EQUAL          =
          0
  599 EA8A4 0000          CON(5) =ZEQ            0=
```

```
          0
600 EA8A9 0000          CON(5) =ZBRNH           IF          ( * NUM * )
          0
601 EA8AE 4100          REL(5) %IF13
          0
602               *
603               * DEVICE SPECIFIED.  SAVE HPIL ADDRESS.
604               *
605 EA8B3 0000          CON(5) =aLSTAD          LISTADDR
          0
606 EA8B8 0000          CON(5) =STORE           !           ( * * )
          0
607 EA8BD 0000          CON(5) =SEMI            ;           ( * * )
          0
608               *
609               * DEVICE SPECIFIED WASN'T PRESENT ON PRIMARY LOOP.
610               *
611 EA8C2         %IF13                         ELSE        ( * NUM * )
612 EA8C2 917A          CON(5) =aLSTAB          LIST.ABORT  ( * * )
          E
613               *
614               * CANNOT CREATE THE RAM FILE SPECIFIED
615               *
616 EA8C7         %IF12                         ELSE        ( * * )
617 EA8C7 0000          CON(5) =LIT             47
          0
618 EA8CC F200          CON(5) 47               'not enough memory for assembler'
          0
619 EA8D1 A07A          CON(5) =aSHUTD          SHUTDOWN
          E
620               *
621               * NO LISTING SPECIFIED
622               *
623 EA8D6         %IF10                         ELSE        ( * ADDR * )
624 EA8D6 0000          CON(5) =DROP            DROP
          0
625 EA8DB 0000          CON(5) =SEMI            ;           ( * * )
          0
626               *
627               * INVALID PAGESIZE (<8)
628               *
629 EA8E0         %IF9                          ELSE
630 EA8E0 0000          CON(5) =LIT             58
          0
631 EA8E5 A300          CON(5) 58               'pagesize too small'
          0
632 EA8EA A07A          CON(5) =aSHUTD          SHUTDOWN
          E
633               ****************************************************************
634               ***
635               ***  SOURCE ABORT
636               ***
637               *** ( -- )
638               *** ( ABORT BECAUSE SOURCE WOULDN'T OPEN )
639               ****************************************************************
```



```
640 EA8EF 0000 =aSRCAB CON(5) =DOCOL          : SOURCE ABORT
          0
641 EA8F4 0000  %IF14   CON(5) =LIT           50
          0
642 EA8F9 2300          CON(5) 50             'cannot open source file'
          0
643 EA8FE A07A          CON(5) =aSHUTD        SHUTDOWN
          E
644               ****************************************************************
645               ***
646               ***  OPEN.SOURCE
647               ***
648               *** ( STR -- )
649               *** ( OPEN SOURCE FILE/SAVE NAME/INITIALIZE VARS )
650               ****************************************************************
651 EA903 0000 =aOPSRC CON(5) =DOCOL         : OPEN.SOURCE
          0
652               *
653               * SAVE OUTER LEVEL SCRFIB, IN CASE IN LOADF
654               *
655 EA908 0000          CON(5) =SCFIB         SCRFIB
          0
656 EA90D 0000          CON(5) =AT            @           ( * STR NUM * )
          0
657 EA912 0000          CON(5) =LIT           SOURCE.FIB#
          0
658 EA917 88CF          CON(5) =oGRAB
          2
659 EA91C 0000          CON(5) =STORE         !
          0
660               *
661               * TRAP ILLEGAL SOURCE FILE NAME
662               *
663 EA921 0000          CON(5) =LIT           ['']
          0
664 EA926 FE8A          CON(5) =aSRCAB        SOURCE ABORT
          E
665 EA92B 0000          CON(5) =ONERR         ONERR
          0
666 EA930 0000          CON(5) =STORE         !
          0
667               *
668               * OPEN SOURCE FILE
669               *
670 EA935 0000          CON(5) =OPNF          OPENF
          0
671 EA93A 0000          CON(5) =ZBRNH         IF          ( * * )
          0
672 EA93F 5BFF          REL(5) %IF14
          F
673               *
674               * RESET TRAP
675               *
676 EA944 0000          CON(5) =ZERO          0
          0
```

```
677 EA949 0000          CON(5) =ONERR           ONERR
          0
678 EA94E 0000          CON(5) =STORE           !
          0
679             *
680             * IS THERE CODE PRESENT IN SOURCE FILE?
681             *
682 EA953 0000          CON(5) =EOF             EOF         ( * F * )
          0
683 EA958 0000          CON(5) =ZBRNH           IF
          0
684 EA95D 4100          REL(5) %IF17
          0
685             *
686             * NO.  MUST ABORT HERE, BECAUSE READ.SOURCE FAILS TO CATCH IT.
687             *
688 EA962 0000          CON(5) =LIT             51
          0
689 EA967 3300          CON(5) 51               'missing/multiple file type'
          0
690 EA96C A07A          CON(5) =aSHUTD          SHUTDOWN
          E
691             *
692             * ALL DONE
693             *
694 EA971        %IF17                          THEN        ( * * )
695 EA971 0000          CON(5) =SEMI            ;           ( * * )
          0
696             ****************************************************************
697             ***
698             ***  RESET.SOURCE
699             ***
700             *** ( -- )
701             *** ( RESET SOURCE FILE TO BEGINNING )
702             ****************************************************************
703 EA976 0000 =aRSSRC CON(5) =DOCOL           : RESET.SOURCE
          0
704 EA97B 0000          CON(5) =ONE             1
          0
705 EA980 0000          CON(5) =BLOCK           BLOCK
          0
706 EA985 0000          CON(5) =DROP            DROP
          0
707 EA98A 0000          CON(5) =SEMI            ;           ( * * )
          0
708             ****************************************************************
709             ***
710             ***  READ.SOURCE
711             ***
712             *** ( -- F )
713             *** ( READ NEXT LINE OF SOURCE, TRUE=NEW LINE READ )
714             ****************************************************************
715 EA98F 0000 =aRDSRC CON(5) =DOCOL           : READ.SOURCE
          0
716             *
```



```
717             * RETURN FF IF WE'RE AT END OF FILE
718             *
719 EA994 0000          CON(5) =EOF             EOF
          0
720 EA999 0000          CON(5) =ZBRNH           IF
          0
721 EA99E F000          REL(5) %IF15
          0
722 EA9A3 0000          CON(5) =ZERO            0
          0
723 EA9A8 0000          CON(5) =SEMI            ;           ( * FF * )
          0
724             *
725             * READ APPROPRIATE SOURCE LINE
726             *
727 EA9AD        %IF15
728 EA9AD 0000          CON(5) =aNEG1           -1          ( * TF * )
          0
729 EA9B2 0000          CON(5) =aSLIN#          SLINE#
          0
730 EA9B7 0000          CON(5) =AT              @
          0
731 EA9BC 0000          CON(5) =BLOCK           BLOCK       ( * TF PTR * )
          0
732             *
733             * INCREMENT SOURCE LINE COUNT
734             *
735 EA9C1 0000          CON(5) =ONE             1
          0
736 EA9C6 0000          CON(5) =aSLIN#          SLINE#
          0
737 EA9CB 0000          CON(5) =PSTOR           +!
          0
738             *
739             * STORE SOURCE STRING, WITH MAX LENGTH 95, IN SLINE
740             *
741 EA9D0 0000          CON(5) =DUP             DUP
          0
742 EA9D5 0000          CON(5) =LIT             4
          0
743 EA9DA 4000          CON(5) 4
          0
744 EA9DF 0000          CON(5) =MINUS           -           ( * TF PTR PTR' * )
          0
745 EA9E4 0000          CON(5) =4N@             4N@
          0
746 EA9E9 0000          CON(5) =LIT             95
          0
747 EA9EE F500          CON(5) 96-1
          0
748 EA9F3 0000          CON(5) =MIN             MIN
          0
749 EA9F8 0000          CON(5) =aSLINE          SLINE
          0
750 EA9FD 0000          CON(5) =S!              S!          ( * TF * )
```

```
          0
   751 EAA02 0000        CON(5) =SEMI          ;          ( * TF * )
          0
   752           ****************************************************************
   753           ***
   754           ***  OBJ.INIT
   755           ***
   756           *** ( -- )
   757           *** ( CREATE AN OBJECT FILE IN THE FILE CHAIN )
   758           ****************************************************************
   759 EAA07 0000 =aOBJIN CON(5) =DOCOL         : OBJ.INIT
          0
   760           *
   761           * GENERATE OBJECT FILE NAME
   762           *
   763 EAA0C 0000        CON(5) =QUOTC         " ASMBOB"   ( * STR * )
          0
   764 EAA11 60          CON(2) 6
   765 EAA13 60          CON(2) 6
   766 EAA15 1435        NIBASC \ASMBOB\
          D424
          F424
   767           *
   768           * SAVE IN OBJECT FILE NAME VARIABLE OFILE
   769           *
   770 EAA21 0000        CON(5) =2DUP          2DUP
          0
   771 EAA26 0000        CON(5) =aOFILE        OFILE
          0
   772 EAA2B 0000        CON(5) =S!            S!          ( * STR * )
          0
   773           *
   774           * CREATE FILE
   775           *
   776 EAA30 0000        CON(5) =ONE           1
          0
   777 EAA35 0000        CON(5) =FCREAT        FCREATE
          0
   778 EAA3A 0000        CON(5) =?DUP          ?DUP
          0
   779 EAA3F 0000        CON(5) =ZBRNH         IF          ( * [ADDR] * )
          0
   780 EAA44 3200        REL(5) %IF16
          0
   781           *
   782           * SET LOCATION COUNTER & OBJECT FILE VARIABLE OBJFILE TO FILE STA
   783           *
   784 EAA49 0000        CON(5) =DUP           DUP
          0
   785 EAA4E 0000        CON(5) =aLC           LC
          0
   786 EAA53 0000        CON(5) =STORE         !           ( * ADDR * )
          0
   787 EAA58 0000        CON(5) =aOBJFL        OBJFILE
          0
```



```
   788 EAA5D 0000        CON(5) =STORE         !           ( * * )
          0
   789 EAA62 0000        CON(5) =SEMI          ;           ( * * )
          0
   790           *
   791           * COULDN'T CREATE OBJECT FILE
   792           *
   793 EAA67      %IF16
   794 EAA67 0000        CON(5) =LIT           47
          0
   795 EAA6C F200        CON(5) 47             'not enough memory for assembler'
          0
   796 EAA71 A07A        CON(5) =aSHUTD        SHUTDOWN    ( * * )
          E
   797           ****************************************************************
   798           ***
   799           ***  OBJ.GROW
   800           ***
   801           *** ( -- )
   802           *** ( GROW THE OBJECT FILE TO THE CORRECT SIZE )
   803           ****************************************************************
   804 EAA76 0000 =aOBJGR CON(5) =DOCOL         : OBJ.GROW
          0
   805           *
   806           * FIND BEGINNING OF OBJECT FILE, PAST HEADER
   807           *
   808 EAA7B 0000        CON(5) =aOFILE        OFILE
          0
   809 EAA80 0000        CON(5) =FFIND         FFIND
          0
   810 EAA85 0000        CON(5) =LIT           37
          0
   811 EAA8A 5200        CON(5) 37
          0
   812 EAA8F 0000        CON(5) =ADD           +           ( * ADDR * )
          0
   813           *
   814           * CALCULATE NUMBER OF BYTES NEEDED TO MAKE FILE CORRECT SIZE
   815           *
   816 EAA94 0000        CON(5) =aFLSIZ        FILESIZE
          0
   817 EAA99 0000        CON(5) =AT            @
          0
   818 EAA9E 0000        CON(5) =LIT           38
          0
   819 EAAA3 6200        CON(5) 38
          0
   820 EAAA8 0000        CON(5) =MINUS         -
          0
   821 EAAAD 0000        CON(5) =ZERO          0
          0
   822 EAAB2 0000        CON(5) =MAX           MAX         ( * ADDR CNT * )
          0
   823           *
   824           * GROW THE OBJECT FILE
```

```
825                 *
826 EAAB7 0000              CON(5) =FADJ            FADJUST     ( * F * )
          0
827 EAABC 0000              CON(5) =ZEQ             0=
          0
828 EAAC1 0000              CON(5) =ZBRNH           IF          ( * * )
          0
829 EAAC6 4100              REL(5) %IF18
          0
830                 *
831                 * COULDN'T GROW IT
832                 *
833 EAACB 0000              CON(5) =LIT             47
          0
834 EAAD0 F200              CON(5) 47               'not enough memory for assembler'
          0
835 EAAD5 A07A              CON(5) =aSHUTD          SHUTDOWN    ( * * )
          E
836                 *
837                 * OBJECT FILE GROWTH CAUSES MEMORY MOVEMENT
838                 *
839 EAADA              %IF18                       THEN
840 EAADA 0000              CON(5) =aMEMOV          MEMMOVE
          0
841 EAADF 0000              CON(5) =SEMI            ;           ( * * )
          0
842                 ****************************************************************
843                 ***
844                 ***  OUT.OBJ
845                 ***
846                 *** ( -- )
847                 *** ( OUTPUT OPCODE TO OBJECT FILE )
848                 ****************************************************************
849 EAAE4 0000 =aOUTOB CON(5) =DOCOL            : OUT.OBJ
          0
850                 *
851                 * NIBBLES TO OUTPUT?
852                 *
853 EAAE9 0000              CON(5) =aOPLEN          OPLEN
          0
854 EAAEE 0000              CON(5) =AT              @
          0
855 EAAF3 0000              CON(5) =?DUP            ?DUP
          0
856 EAAF8 0000              CON(5) =ZBRNH           IF          ( * NUM * )
          0
857 EAAFD 3700              REL(5) %IF19
          0
858                 *
859                 * LOOP THROUGH ALL NIBBLES TO OUTPUT
860                 *
861 EAB02 0000              CON(5) =ZERO            0
          0
862 EAB07 0000              CON(5) =XDO             DO
          0
```



```
863 EAB0C           %DO2                                    ( * * )
864                 *
865                 * IF WE'RE AT THE 18TH (OR GREATER) NIBBLE, THEN IT WAS GENERATED
866                 * BY THE BSS PSEUDO-OP, AND MUST BE ZERO
867                 *
868 EAB0C 0000              CON(5) =R@              I
          0
869 EAB11 0000              CON(5) =LIT             17
          0
870 EAB16 1100              CON(5) 17
          0
871 EAB1B 0000              CON(5) =GT              >
          0
872 EAB20 0000              CON(5) =ZBRNH           IF          ( * * )
          0
873 EAB25 4100              REL(5) %IF20
          0
874 EAB2A 0000              CON(5) =ZERO            0           ( * NIB * )
          0
875 EAB2F 0000              CON(5) =BRNCH           ELSE
          0
876 EAB34 9100              REL(5) %TH20
          0
877 EAB39           %IF20                                   ( * * )
878                 *
879                 * OTHERWISE, FETCH APPROPRIATE NIBBLE TO STORE
880                 *
881 EAB39 0000              CON(5) =R@              I
          0
882 EAB3E 0000              CON(5) =ONE+            1+
          0
883 EAB43 3C4A              CON(5) =a>OPx           >OPx
          E
884 EAB48 0000              CON(5) =AT              @           ( * NIB * )
          0
885                 *
886                 * STORE NIBBLE AT LOCATION COUNTER
887                 *
888 EAB4D           %TH20                           THEN        ( * NIB * )
889 EAB4D 0000              CON(5) =aLC             LC
          0
890 EAB52 0000              CON(5) =AT              @
          0
891 EAB57 0000              CON(5) =R@              I
          0
892 EAB5C 0000              CON(5) =ADD             +
          0
893 EAB61 0000              CON(5) =N!              N!          ( * * )
          0
894 EAB66 0000              CON(5) =XLOOP           LOOP        ( * * )
          0
895 EAB6B 1AFF              REL(5) %DO2
          F
896                 *
897                 * ALL DONE
```

```
898             *
899 EAB70        %IF19                      THEN
900 EAB70 0000          CON(5) =SEMI       ;          ( * * )
        0
901             ***************************************************************
902             ***
903             ***  EJECT
904             ***
905             *** ( -- )
906             *** ( EJECT NEW PAGE )
907             ***************************************************************
908 EAB75 0000 =aEJECT CON(5) =DOCOL        : EJECT
        0
909             *
910             * ARE WE LISTING AT THE MOMENT?  (LIST ON?)
911             *
912 EAB7A 0000          CON(5) =aTOLST       TOLIST
        0
913 EAB7F 0000          CON(5) =AT           @
        0
914 EAB84 0000          CON(5) =ZBRNH        IF         ( * * )
        0
915 EAB89 AD00          REL(5) %IF21
        0
916             *
917             * YES.  START A NEW LINE COUNT
918             *
919 EAB8E 0000          CON(5) =ZERO         0
        0
920 EAB93 0000          CON(5) =aLLIN#       LLINE#
        0
921 EAB98 0000          CON(5) =STORE        !          ( * * )
        0
922 EAB9D 0000          CON(5) =ONE          1
        0
923 EABA2 0000          CON(5) =aPAGE#       PAGENO
        0
924 EABA7 0000          CON(5) =PSTOR        +!         ( * * )
        0
925             *
926             * OUT A FORM FEED
927             *
928 EABAC 0000          CON(5) =LIT          12
        0
929 EABB1 C000          CON(5) 12
        0
930 EABB6 0000          CON(5) =CHR$         CHR$
        0
931 EABBB 99DA          CON(5) =aOUTLI       OUT.LINE   ( * * )
        E
932             *
933             * GENERATE PAGE xxx STRING
934             *
935 EABC0 0000          CON(5) =aLLINE       LLINE
        0
```



```
936 EABC5 0000          CON(5) =QUOTC        " PAGE "
        0
937 EABCA 50            CON(2) 5
938 EABCC 50            CON(2) 5
939 EABCE 0514          NIBASC \PAGE \
        7454
        02
940 EABD8 0000          CON(5) =S<&          S<&        ( * STR * )
        0
941 EABDD 0000          CON(5) =aPAGE#       PAGENO
        0
942 EABE2 0000          CON(5) =AT           @
        0
943 EABE7 0000          CON(5) =DEC          DECIMAL
        0
944 EABEC 054A          CON(5) =aNUM#3       #3
        E
945 EABF1 0000          CON(5) =S<&          S<&        ( * STR * )
        0
946             *
947             * APPEND TITLE TO PAGE xxx STRING & OUTPUT LINE
948             *
949 EABF6 003A          CON(5) =aBLNKB       11BLANK    ( * STR STR' * )
        E
950 EABFB 0000          CON(5) =S<&          S<&        ( * STR * )
        0
951 EAC00 0000          CON(5) =aTIT         TIT
        0
952 EAC05 0000          CON(5) =S<&          S<&
        0
953 EAC0A 99DA          CON(5) =aOUTLI       OUT.LINE   ( * * )
        E
954             *
955             * GENERATE Forth Assembler STRING
956             *
957 EAC0F 0000          CON(5) =aLLINE       LLINE
        0
958 EAC14 0000          CON(5) =QUOTC        " Forth Assembler     "
        0
959 EAC19 41            CON(2) 20
960 EAC1B 41            CON(2) 20
961 EAC1D 64F6          NIBASC \Forth As\
        2747
        8602
        1437
962 EAC2D 3756          NIBASC \sembler \
        D626
        C656
        2702
963 EAC3D 0202          NIBASC \    \
        0202
964 EAC45 0000          CON(5) =S<&          S<&        ( * STR * )
        0
965             *
966             * APPEND SUB TITLE TO Forth Assembler STRING & OUTPUT LINE
```

```
 967                *
 968 EAC4A 0000     0       CON(5) =aSTIT           STIT
 969 EAC4F 0000     0       CON(5) =S<&             S<&         ( * STR * )
 970 EAC54 99DAE    0       CON(5) =aOUTLI          OUT.LINE    ( * * )
 971                *
 972                * OUTPUT A BLANK LINE
 973                *
 974 EAC59 0000     0       CON(5) =NULL$           NULL$
 975 EAC5E 99DAE    0       CON(5) =aOUTLI          OUT.LINE    ( * * )
 976                *
 977                * ALL DONE
 978                *
 979 EAC63                  %IF21                   THEN
 980 EAC63 0000     0       CON(5) =SEMI            ;           ( * * )
 981                ****************************************************************
 982                ***
 983                ***  PRINT#
 984                ***
 985                *** ( STR -- )
 986                *** ( APPEND STRING TO END OF LISTING FILE )
 987                ****************************************************************
 988 EAC68 0000     0 =aPRT#  CON(5) =DOCOL         : PRINT#
 989                *
 990                * SAVE STRING IN VARIABLE LLINE (WE USE THE PAD!)
 991                *
 992 EAC6D 0000     0       CON(5) =aLLINE          LLINE
 993 EAC72 0000     0       CON(5) =S!              S!
 994 EAC77 0000     0       CON(5) =aLLINE          LLINE       ( * STR * )
 995                *
 996                * DO WE HAVE AN EVEN LENGTH STRING?
 997                *
 998 EAC7C 0000     0       CON(5) =DUP             DUP
 999 EAC81 0000     0       CON(5) =DUP             DUP
1000 EAC86 0000     0       CON(5) =TWO/            2/
1001 EAC8B 0000     0       CON(5) =TWO*            2*
1002 EAC90 0000     0       CON(5) =EQUAL           =
1003 EAC95 0000     0       CON(5) =ZEQ             0=
1004 EAC9A 0000     0       CON(5) =ZBRNH           IF          ( * STR * )
```



```
1005 EAC9F 4100     0       REL(5) %IF22
1006                *
1007                * NO, TACK AN ASCII 0 ON THE END TO MAKE IT EVEN (LIF1 STANDARD)
1008                *
1009 EACA4 0000     0       CON(5) =ZERO            0
1010 EACA9 0000     0       CON(5) =CHR$            CHR$
1011 EACAE 0000     0       CON(5) =S<&             S<&         ( * STR * )
1012                *
1013                * CALCULATE ADDRESS TO PUT STRING
1014                *
1015 EACB3                  %IF22                   THEN        ( * STR * )
1016 EACB3 0000     0       CON(5) =SWAP            SWAP
1017 EACB8 0000     0       CON(5) =DROP            DROP        ( * LEN * )
1018 EACBD 0000     0       CON(5) =aLISTF          LISTFILE
1019 EACC2 0000     0       CON(5) =AT              @
1020 EACC7 0000     0       CON(5) =LIT             37
1021 EACCC 5200     0       CON(5) 37
1022 EACD1 0000     0       CON(5) =ADD             +           ( * LEN ADDR * )
1023 EACD6 0000     0       CON(5) =aLNOFF          LNOFF
1024 EACDB 0000     0       CON(5) =AT              @
1025 EACE0 0000     0       CON(5) =ADD             +           ( * LEN ADDR * )
1026                *
1027                * ADJUST LISTING FILE TO FIT NEW STRING
1028                *
1029 EACE5 0000     0       CON(5) =OVER            OVER
1030 EACEA 0000     0       CON(5) =TWO*            2*
1031 EACEF 0000     0       CON(5) =FOUR+           4+          ( * LEN ADDR CNT * )
1032 EACF4 0000     0       CON(5) =2DUP            2DUP
1033 EACF9 0000     0       CON(5) =FADJ            FADJUST
1034 EACFE 0000     0       CON(5) =ZBRNH           IF          ( * LEN ADDR CNT * )
1035 EAD03 B400     0       REL(5) %IF23
1036                *
1037                * SUCCESS.  UPDATE OFFSET FOR NEXT LINE
1038                *
```

```
1039 EAD08 0000          CON(5) =aLNOFF        LNOFF
           0
1040 EAD0D 0000          CON(5) =PSTOR         +!          ( * LEN ADDR * )
           0
1041             *
1042             * STORE LENGTH (FOUR NIBBLES)
1043             *
1044 EAD12 0000          CON(5) =ZERO          0
           0
1045 EAD17 0000          CON(5) =OVER          OVER
           0
1046 EAD1C 0000          CON(5) =C!            C!
           0
1047 EAD21 0000          CON(5) =TWO+          2+          ( * LEN ADDR * )
           0
1048 EAD26 0000          CON(5) =SWAP          SWAP
           0
1049 EAD2B 0000          CON(5) =OVER          OVER
           0
1050 EAD30 0000          CON(5) =C!            C!
           0
1051 EAD35 0000          CON(5) =TWO+          2+          ( * ADDR * )
           0
1052             *
1053             * MOVE LISTING STRING INTO FILE
1054             *
1055 EAD3A 0000          CON(5) =aLLINE        LLINE
           0
1056 EAD3F 0000          CON(5) =ROT           ROT
           0
1057 EAD44 0000          CON(5) =SMOVE         SMOVE       ( * * )
           0
1058 EAD49 0000          CON(5) =SEMI          ;           ( * * )
           0
1059             *
1060             * COULDN'T GROW FILE FOR NEW STRING
1061             *
1062 EAD4E        %IF23
1063 EAD4E 0000          CON(5) =DROP          DROP
           0
1064 EAD53 0000          CON(5) =2DROP         2DROP       ( * * )
           0
1065             *
1066             * HAVE WE ALREADY ISSUED THE listing file full MESSAGE?
1067             *
1068 EAD58 0000          CON(5) =ONE           1
           0
1069 EAD5D CE3A          CON(5) =aTESTF        TEST.FLAG   ( * F * )
           E
1070 EAD62 0000          CON(5) =ZEQ           0=
           0
1071 EAD67 0000          CON(5) =ZBRNH         IF          ( * * )
           0
1072 EAD6C 8200          REL(5) %IF24
           0
```



```
1073             *
1074             * NO, LET'S ISSUE IT.
1075             *
1076 EAD71 0000          CON(5) =ONE           1
           0
1077 EAD76 9C3A          CON(5) =aSETF         SET.FLAG
           E
1078 EAD7B 0000          CON(5) =LIT           44
           0
1079 EAD80 C200          CON(5) 44             'listing file full'
           0
1080 EAD85 0000          CON(5) =aERMSG        ERRMSG
           0
1081 EAD8A 0000          CON(5) =BRNCH         GOTO
           0
1082 EAD8F 0E30          REL(5) %TYPE
           0
1083             *
1084             * ALL DONE
1085             *
1086 EAD94        %IF24                        THEN
1087 EAD94 0000          CON(5) =SEMI          ;           ( * * )
           0
1088             ****************************************************************
1089             ***
1090             ***  OUT.LINE
1091             ***
1092             *** ( STR -- )
1093             *** ( OUTPUT THE STRING TO LISTING FILE/DEVICE )
1094             ****************************************************************
1095 EAD99 0000 =aOUTLI CON(5) =DOCOL          : OUT.LINE
           0
1096             *
1097             * CHECK FOR ATTN KEY
1098             *
1099 EAD9E 0000          CON(5) =aMONAT        MONITOR.ATTN
           0
1100             *
1101             * ARE WE LISTING NOW?  (LIST ON? AND LISTING SPECIFIED)
1102             *
1103 EADA3 0000          CON(5) =aTOLST        TOLIST
           0
1104 EADA8 0000          CON(5) =AT            @           ( * F * )
           0
1105 EADAD 0000          CON(5) =LISTNG        LISTING
           0
1106 EADB2 0000          CON(5) =SWAP          SWAP
           0
1107 EADB7 0000          CON(5) =DROP          DROP        ( * F LEN * )
           0
1108 EADBC 0000          CON(5) =F.AND         F.AND       ( * F * )
           0
1109 EADC1 0000          CON(5) =ZBRNH         IF          ( * STR * )
           0
1110 EADC6 2800          REL(5) %IF300
```

```
          0
1111            *
1112            * ARE WE SENDING THE LISTING TO AN HPIL DEVICE?
1113            *
1114 EADCB 0000          CON(5) =aLSTAD          LISTADDR
          0
1115 EADD0 0000          CON(5) =AT              @
          0
1116 EADD5 0000          CON(5) =?DUP            ?DUP          ( * STR [NUM] * )
          0
1117 EADDA 0000          CON(5) =ZBRNH           IF
          0
1118 EADDF 8200          REL(5) %IF301
          0
1119            *
1120            * YES, OUTPUT LINE
1121            *
1122 EADE4 0000          CON(5) =PRIME           PRIMARY
          0
1123 EADE9 0000          CON(5) =STORE           !             ( * STR * )
          0
1124 EADEE 0000          CON(5) =OUTPUT          OUTUPT
          0
1125 EADF3 0000          CON(5) =CRLF            CRLF
          0
1126 EADF8 0000          CON(5) =OUTPUT          OUTPUT        ( * * )
          0
1127 EADFD 0000          CON(5) =BRNCH           ELSE          ( * STR * )
          0
1128 EAE02 A000          REL(5) %TH301
          0
1129            *
1130            * NO, APPEND LINE TO LISTING FILE
1131            *
1132 EAE07       %IF301
1133 EAE07 86CA          CON(5) =aPRT#           PRINT#
          E
1134            *
1135            * UPDATE LINES THIS PAGE
1136            *
1137 EAE0C       %TH301                          THEN          ( * * )
1138 EAE0C 0000          CON(5) =ONE             1
          0
1139 EAE11 0000          CON(5) =aLLIN#          LLINE#
          0
1140 EAE16 0000          CON(5) =PSTOR           +!            ( * * )
          0
1141            *
1142            * ARE WE AT THE END OF A PAGE?
1143            *
1144 EAE1B 0000          CON(5) =aLLIN#          LLINE#
          0
1145 EAE20 0000          CON(5) =AT              @
          0
1146 EAE25 0000          CON(5) =PAGESZ          PAGESIZE
```



```
          0
1147 EAE2A 0000          CON(5) =AT              @             ( * NUM NUM' * )
          0
1148 EAE2F 0000          CON(5) =GT              >
          0
1149 EAE34 0000          CON(5) =ZBRNH           IF
          0
1150 EAE39 A000          REL(5) %IF303
          0
1151            *
1152            * YES.  DO A FORM FEED.
1153            *
1154 EAE3E 57BA          CON(5) =aEJECT          EJECT         ( * * )
          E
1155            *
1156            * CLEAR STACK AND VARIABLE LLINE
1157            *
1158 EAE43       %IF303                          THEN          ( * * )
1159 EAE43 0000          CON(5) =NULL$           NULL$         ( * STR * )
          0
1160 EAE48       %IF300
1161 EAE48 0000          CON(5) =2DROP           2DROP
          0
1162 EAE4D 0B3A          CON(5) =a&CLR           &CLEAR
          E
1163 EAE52 0000          CON(5) =SEMI            ;             ( * * )
          0
1164            ****************************************************************
1165            ***
1166            ***  OUT.LIST
1167            ***
1168            *** ( -- )
1169            *** ( OUTPUT CURRENT LINE TO LISTING FILE )
1170            ****************************************************************
1171 EAE57 0000 =aOUTLS CON(5) =DOCOL           : OUT.LIST
          0
1172            *
1173            * ARE WE LISTING NOW?   (LIST ON? AND LISTING SPECIFIED)
1174            *
1175 EAE5C 0000          CON(5) =LISTNG          LISTING
          0
1176 EAE61 0000          CON(5) =SWAP            SWAP
          0
1177 EAE66 0000          CON(5) =DROP            DROP          ( * LEN * )
          0
1178 EAE6B 0000          CON(5) =aTOLST          TOLIST
          0
1179 EAE70 0000          CON(5) =AT              @             ( * LEN F * )
          0
1180 EAE75 0000          CON(5) =F.AND           F.AND
          0
1181 EAE7A 0000          CON(5) =ZBRNH           IF            ( * * )
          0
1182 EAE7F 9F10          REL(5) %IF302
          0
```

```
1183            *
1184            * SET UP A LOOP FOR EACH LINE ON THE OUTPUT LISTING.  EACH LINE C
1185            * CONTAIN NO MORE THAN EIGHT (8) NIBBLES OF OPCODE.  NO OPCODE
1186            * (INCLUDING BSS) CAN CREATE MORE THAN THREE LINES OF LISTING
1187            *
1188 EAE84 0000         CON(5) =aOPLEN        OPLEN
         0
1189 EAE89 0000         CON(5) =AT            @
         0
1190 EAE8E 0000         CON(5) =LIT           24
         0
1191 EAE93 8100         CON(5) 24
         0
1192 EAE98 0000         CON(5) =MIN           MIN         ( * NUM * )
         0
1193 EAE9D 0000         CON(5) =LIT           7
         0
1194 EAEA2 7000         CON(5) 7
         0
1195 EAEA7 0000         CON(5) =ADD           +
         0
1196 EAEAC 0000         CON(5) =LIT           8
         0
1197 EAEB1 8000         CON(5) 8
         0
1198 EAEB6 0000         CON(5) =/             /
         0
1199 EAEBB 0000         CON(5) =ONE           1
         0
1200 EAEC0 0000         CON(5) =MAX           MAX
         0
1201 EAEC5 0000         CON(5) =ZERO          0
         0
1202 EAECA 0000         CON(5) =XDO           DO          ( * * )
         0
1203 EAECF       %DO3
1204            *
1205            * CLEAR VARIABLE LLINE TO START OUR BUILDING OF LISTING LINE.
1206            *
1207 EAECF 0B3A         CON(5) =a&CLR         &CLEAR
         E
1208 EAED4 0000         CON(5) =aLLINE        LLINE       ( * STR * )
         0
1209 EAED9 0000         CON(5) =R@            I
         0
1210 EAEDE 0000         CON(5) =ZEQ           0=
         0
1211 EAEE3 0000         CON(5) =ZBRNH         IF          ( * STR * )
         0
1212 EAEE8 8200         REL(5) %IF29
         0
1213            *
1214            * FIRST LINE.  GENERATE LINE NUMBER.
1215            *
1216 EAEED 0000         CON(5) =DEC           DECIMAL
```



```
         0
1217 EAEF2 0000         CON(5) =aSLIN#        SLINE#
         0
1218 EAEF7 0000         CON(5) =AT            @           ( * STR NUM * )
         0
1219 EAEFC 0000         CON(5) =ONE-          1-
         0
1220 EAF01 E64A         CON(5) =aNUM#4        #4          ( * STR STR' * )
         E
1221 EAF06 0000         CON(5) =BRNCH         ELSE        ( * STR * )
         0
1222 EAF0B A000         REL(5) %TH29
         0
1223            *
1224            * NOT FIRST LINE.  GENERATE BLANKS.
1225            *
1226 EAF10       %IF29
1227 EAF10 923A         CON(5) =aBLNK5        "     "     ( * STR STR' * )
         E
1228 EAF15       %TH29                        THEN        ( * STR STR' * )
1229 EAF15 0000         CON(5) =S<&           S<&         ( * STR * )
         0
1230            *
1231            * GENERATE LOCATION COUNTER
1232            *
1233 EAF1A 0000         CON(5) =HEX           HEX
         0
1234 EAF1F 0000         CON(5) =aLC           LC
         0
1235 EAF24 0000         CON(5) =AT            @
         0
1236 EAF29 0000         CON(5) =R@            I
         0
1237 EAF2E 0000         CON(5) =LIT           8
         0
1238 EAF33 8000         CON(5) 8
         0
1239 EAF38 0000         CON(5) =MULT          *
         0
1240 EAF3D 0000         CON(5) =ADD           +           ( * STR NUM * )
         0
1241            *
1242            * CHECK FOR LEX OR BIN FILE TYPE
1243            *
1244 EAF42 0000         CON(5) =aFLTYP        FILETYPE
         0
1245 EAF47 0000         CON(5) =AT            @           ( * STR NUM TYP * )
         0
1246 EAF4C 0000         CON(5) =ONE           1
         0
1247 EAF51 0000         CON(5) =GT            >
         0
1248 EAF56 0000         CON(5) =ZBRNH         IF          ( * STR NUM * )
         0
1249 EAF5B 4100         REL(5) %IF305
```

```
                0
1250                    *
1251                    * LOCATION COUNTER IS RELATIVE TO START OF FILE.
1252                    *
1253 EAF60 0000                CON(5) =aOBJFL          OBJFILE
                0
1254 EAF65 0000                CON(5) =AT              @
                0
1255 EAF6A 0000                CON(5) =MINUS           -
                0
1256 EAF6F              %IF305                          THEN            ( * STR NUM * )
1257 EAF6F C84A                CON(5) =aNUM#5          #5
                E
1258 EAF74 0000                CON(5) =S<&             S<&             ( * STR * )
                0
1259 EAF79 0000                CON(5) =R@              I
                0
1260                    *
1261                    * SET UP A LOOP FOR 8 NIBBLES OF OPCODE
1262                    *
1263 EAF7E 0000                CON(5) =LIT             8
                0
1264 EAF83 8000                CON(5) 8
                0
1265 EAF88 0000                CON(5) =MULT            *
                0
1266 EAF8D 0000                CON(5) =DUP             DUP
                0
1267 EAF92 0000                CON(5) =LIT             8
                0
1268 EAF97 8000                CON(5) 8
                0
1269 EAF9C 0000                CON(5) =ADD             +
                0
1270 EAFA1 0000                CON(5) =SWAP            SWAP
                0
1271 EAFA6 0000                CON(5) =XDO             DO              ( * STR * )
                0
1272 EAFAB              %DO4
1273                    *
1274                    * HAVE WE REACHED THE END OF THIS OPCODE?
1275                    *
1276 EAFAB 0000                CON(5) =R@              I
                0
1277 EAFB0 0000                CON(5) =aOPLEN          OPLEN
                0
1278 EAFB5 0000                CON(5) =AT              @
                0
1279 EAFBA 0000                CON(5) =ONE-            1-
                0
1280 EAFBF 0000                CON(5) =GT              >
                0
1281 EAFC4 0000                CON(5) =ZBRNH           IF              ( * STR * )
                0
1282 EAFC9 4100                REL(5) %IF30
```



```
                0
1283                    *
1284                    * YES.  GENERATE BLANKS.
1285                    *
1286 EAFCE 063A                CON(5) =aBLNK1          1BLANK          ( * STR STR' * )
                E
1287 EAFD3 0000                CON(5) =BRNCH           ELSE            ( * STR * )
                0
1288 EAFD8 B400                REL(5) %TH30
                0
1289                    *
1290                    * NO.  GET NIBBLE FROM VARIABLE LOCATIONS.
1291                    *
1292 EAFDD              %IF30
1293 EAFDD 0000                CON(5) =R@              I
                0
1294 EAFE2 0000                CON(5) =LIT             17
                0
1295 EAFE7 1100                CON(5) 17
                0
1296 EAFEC 0000                CON(5) =GT              >
                0
1297 EAFF1 0000                CON(5) =ZBRNH           IF              ( * STR * )
                0
1298 EAFF6 4100                REL(5) %IF31
                0
1299                    *
1300                    * LENGTH > 17.  MUST HAVE BEEN GENEREATED BY THE BSS PSEUDO-OP.
1301                    * HENCE, MUST BE ZERO.
1302                    *
1303 EAFFB 0000                CON(5) =ZERO            0               ( * STR NUM * )
                0
1304 EB000 0000                CON(5) =BRNCH           ELSE            ( * STR * )
                0
1305 EB005 9100                REL(5) %TH31
                0
1306 EB00A              %IF31
1307 EB00A 0000                CON(5) =R@              I
                0
1308 EB00F 0000                CON(5) =ONE+            1+
                0
1309 EB014 3C4A                CON(5) =a>OPx           >OPx
                E
1310 EB019 0000                CON(5) =AT              @               ( * STR NUM * )
                0
1311 EB01E              %TH31                           THEN
1312 EB01E E14A                CON(5) =aNUM#1          #1              ( * STR STR' * )
                E
1313                    *
1314                    * APPEND NIBBLE TO OUTPUT STRING
1315                    *
1316 EB023              %TH30                           THEN
1317 EB023 0000                CON(5) =S<&             S<&             ( * STR * )
                0
1318 EB028 0000                CON(5) =XLOOP           LOOP            ( * STR * )
```

```
          0
1319 EB02D E7FF         REL(5) %DO4
          F
1320            *
1321            * FIRST LINE?
1322            *
1323 EB032 0000         CON(5) =R@              I
          0
1324 EB037 0000         CON(5) =ZEQ             0=
          0
1325 EB03C 0000         CON(5) =ZBRNH           IF          ( * STR * )
          0
1326 EB041 8200         REL(5) %IF32
          0
1327            *
1328            * YES.  APPEND SOURCE LINE (ONLY 75 CHARS) TO LISTING LINE
1329            *
1330 EB046 743A         CON(5) =aBLNK2          2BLANK
          E
1331 EB04B 0000         CON(5) =S<&             S<&         ( * STR * )
          0
1332 EB050 0000         CON(5) =aSLINE          SLINE
          0
1333 EB055 0000         CON(5) =LIT             75
          0
1334 EB05A B400         CON(5) 75
          0
1335 EB05F 0000         CON(5) =MIN             MIN
          0
1336 EB064 0000         CON(5) =S<&             S<&         ( * STR * )
          0
1337            *
1338            * OUTPUT LINE
1339            *
1340 EB069       %IF32                          THEN
1341 EB069 99DA         CON(5) =aOUTLI          OUT.LINE    ( * * )
          E
1342 EB06E 0000         CON(5) =XLOOP           LOOP
          0
1343 EB073 C5EF         REL(5) %DO3
          F
1344            *
1345            * ALL DONE
1346            *
1347 EB078       %IF302
1348 EB078 0000         CON(5) =DEC             DECIMAL
          0
1349 EB07D 0000         CON(5) =SEMI            ;           ( * * )
          0
1350            ****************************************************************
1351            ***
1352            ***  ERROR
1353            ***
1354            *** ( NUM -- )
1355            *** ( REPORT AN ERROR CORRECTLY )
```



```
1356            ****************************************************************
1357 EB082 0000 =aERROR CON(5) =DOCOL           : ERROR
          0
1358            *
1359            * INCREMENT ERROR COUNT
1360            *
1361 EB087 0000         CON(5) =ONE             1
          0
1362 EB08C 0000         CON(5) =aERCNT          ERROR COUNT
          0
1363 EB091 0000         CON(5) =PSTOR           +!
          0
1364            *
1365            * ALL ERROR LINES BEGIN WITH A "*"
1366            *
1367 EB096 0000         CON(5) =QUOTC           " * "       ( * NUM STR * )
          0
1368 EB09B 20           CON(2) 2
1369 EB09D 20           CON(2) 2
1370 EB09F A202         NIBASC \* \
1371            *
1372            * APPEND ERROR MESSAGE TO "*"
1373            *
1374 EB0A3 0000         CON(5) =ROT             ROT         ( * STR NUM * )
          0
1375 EB0A8 0000         CON(5) =aERMSG          ERRMSG      ( * STR STR' * )
          0
1376 EB0AD 0000         CON(5) =S>&             S>&         ( * STR * )
          0
1377            *
1378            * SAVE ERROR MESSAGE
1379            *
1380 EB0B2 0000         CON(5) =aERR$           ERR$
          0
1381 EB0B7 0000         CON(5) =S!              S!          ( * * )
          0
1382            *
1383            * ARE WE LISTING NOW?  (LIST ON? AND LISTING SPECIFIED)
1384            *
1385 EB0BC 0000         CON(5) =aTOLST          TOLIST
          0
1386 EB0C1 0000         CON(5) =AT              @           ( * NUM * )
          0
1387 EB0C6 0000         CON(5) =LISTNG          LISTING
          0
1388 EB0CB 0000         CON(5) =SWAP            SWAP
          0
1389 EB0D0 0000         CON(5) =DROP            DROP        ( * NUM LEN * )
          0
1390 EB0D5 0000         CON(5) =F.AND           F.AND       ( * F * )
          0
1391 EB0DA 0000         CON(5) =ZBRNH           IF          ( * * )
          0
1392 EB0DF B400         REL(5) %IF33
          0
```

```
1393                *
1394                * HAVE WE ISSUED AN ERROR ALREADY THIS LINE?
1395                *
1396 EB0E4 0000          CON(5) =LIT          8
             0
1397 EB0E9 8000          CON(5) 8
             0
1398 EB0EE CE3A          CON(5) =aTESTF       TEST.FLAG    ( * F * )
             E
1399 EB0F3 0000          CON(5) =ZEQ          0=
             0
1400 EB0F8 0000          CON(5) =ZBRNH        IF           ( * * )
             0
1401 EB0FD E100          REL(5) %IF34
             0
1402                *
1403                * NO.  ISSUE A BLANK LINE FIRST TO DELIMIT IT.
1404                *
1405 EB102 0000          CON(5) =LIT          8
             0
1406 EB107 8000          CON(5) 8
             0
1407 EB10C 9C3A          CON(5) =aSETF        SET.FLAG     ( * * )
             E
1408 EB111 0000          CON(5) =NULL$        NULL$
             0
1409 EB116 99DA          CON(5) =aOUTLI       OUT.LINE     ( * * )
             E
1410                *
1411                * OUTPUT ERROR
1412                *
1413 EB11B           %IF34                     THEN
1414 EB11B 0000          CON(5) =aERR$        ERR$
             0
1415 EB120 99DA          CON(5) =aOUTLI       OUT.LINE     ( * * )
             E
1416 EB125 0000          CON(5) =SEMI         ;            ( * * )
             0
1417                *
1418                * NOT LISTING NOW, SO SEND ERROR TO DISPLAY
1419                *
1420 EB12A           %IF33                                  ( * * )
1421 EB12A 0000          CON(5) =aERR$        ERR$         ( * STR * )
             0
1422                *
1423                * APPEND A , in line xxxx STRING TO ERROR MESSAGE
1424                *
1425 EB12F 0000          CON(5) =QUOTC        " , in line "
             0
1426 EB134 A0            CON(2) 10
1427 EB136 A0            CON(2) 10
1428 EB138 C202          NIBASC \, in lin\
             96E6
             02C6
             96E6
```



```
1429 EB148 5602          NIBASC \e \
1430 EB14C C93A          CON(5) =a>PAD        >PAD         ( * STR STR' * )
             E
1431 EB151 0000          CON(5) =S>&          S>&          ( * STR * )
             0
1432 EB156 0000          CON(5) =aSLIN#       SLINE#
             0
1433 EB15B 0000          CON(5) =AT           @            ( * STR NUM * )
             0
1434 EB160 0000          CON(5) =ONE-         1-           ( * STR NUM * )
             0
1435 EB165 E64A          CON(5) =aNUM#4       #4
             E
1436 EB16A 0000          CON(5) =S<&          S<&          ( * STR * )
             0
1437                *
1438                * OUTPUT STRING AND ZERO LINECNT (VARIABLE USED IN SHOWP)
1439                *
1440 EB16F           %TYPE                     THEN         ( * STR * )
1441 EB16F 0000          CON(5) =CR           CR
             0
1442 EB174 0000          CON(5) =TYPE         TYPE
             0
1443 EB179 0000          CON(5) =ZERO         0
             0
1444 EB17E 0000          CON(5) =aLINEC       LINECNT
             0
1445 EB183 0000          CON(5) =STORE        !            ( * * )
             0
1446 EB188 0000          CON(5) =SEMI         ;            ( * * )
             0
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER

```
1447                    STITLE FIT STATS
1448            zSIZE   EQU    (*)-zTHIS
1449            zLEFT   EQU    (zNEXT)-*
1450 EB18D              BSS    zLEFT
1451 EB200              END
```



```
%DO2    Abs  961292 #EAB0C -   863   895
%DO3    Abs  962255 #EAECF -  1203  1343
%DO4    Abs  962475 #EAFAB -  1272  1319
%IF10   Abs  960726 #EA8D6 -   623   509
%IF11   Abs  960521 #EA809 -   554   516
%IF12   Abs  960711 #EA8C7 -   616   525
%IF13   Abs  960706 #EA8C2 -   611   583   601
%IF14   Abs  960756 #EA8F4 -   641   672
%IF15   Abs  960941 #EA9AD -   727   721
%IF16   Abs  961127 #EAA67 -   793   780
%IF17   Abs  960881 #EA971 -   694   684
%IF18   Abs  961242 #EAADA -   839   829
%IF19   Abs  961392 #EAB70 -   899   857
%IF20   Abs  961337 #EAB39 -   877   873
%IF21   Abs  961635 #EAC63 -   979   915
%IF22   Abs  961715 #EACB3 -  1015  1005
%IF23   Abs  961870 #EAD4E -  1062  1035
%IF24   Abs  961940 #EAD94 -  1086  1072
%IF29   Abs  962320 #EAF10 -  1226  1212
%IF30   Abs  962525 #EAFDD -  1292  1282
%IF300  Abs  962120 #EAE48 -  1160  1116
%IF301  Abs  962055 #EAE07 -  1132  1118
%IF302  Abs  962680 #EB078 -  1347  1182
%IF303  Abs  962115 #EAE43 -  1158  1150
%IF304  Abs  960206 #EA6CE -   430   427
%IF305  Abs  962415 #EAF6F -  1256  1249
%IF31   Abs  962570 #EB00A -  1306  1298
%IF32   Abs  962665 #EB069 -  1340  1326
%IF33   Abs  962858 #EB12A -  1420  1392
%IF34   Abs  962843 #EB11B -  1413  1401
%IF6    Abs  959914 #EA5AA -   310   282
%IF7    Abs  959899 #EA59B -   303   293
%IF8    Abs  960256 #EA700 -   450   444
%IF9    Abs  960736 #EA8E0 -   629   491
%NUM1   Abs  959668 #EA4B4 -   214   160
%NUM2   Abs  959663 #EA4AF -   213   172
%NUM3   Abs  959658 #EA4AA -   212   185
%NUM4   Abs  959653 #EA4A5 -   211   198
%TH20   Abs  961357 #EAB4D -   888   876
%TH29   Abs  962325 #EAF15 -  1228  1222
%TH30   Abs  962595 #EB023 -  1316  1288
%TH301  Abs  962060 #EAE0C -  1137  1128
%TH31   Abs  962590 #EB01E -  1311  1305
%TYPE   Abs  962927 #EB16F -  1440  1082
/       Ext              -  1198
2DROP   Ext              -    85    86   304   313   431  1064  1161
2DUP    Ext              -   278   428   513   540   770  1032
4N@     Ext              -   289   745
>R      Ext              -   277
?DUP    Ext              -   280   507   523   778   855  1116
ABORT   Ext              -   452
ADD     Ext              -   230   288   539   812   892  1022  1025  1195
                         1240  1269
AND     Ext              -   135
AT      Ext              -   120   134   340   347   354   364   486   656
```

```
                          730   817   854   884   890   913   942  1019
                         1024  1104  1115  1145  1147  1179  1189  1218
                         1235  1245  1254  1278  1310  1386  1433
BASE     Ext           -  365
BASF     Ext           -  575
BASICX   Ext           -  257
BLOCK    Ext           -  705   731
BRNCH    Ext           -  159   171   184   197   875  1081  1127  1221
                         1287  1304
C!       Ext           -  541   543  1046  1050
CHR$     Ext           -  930  1010
CLOSF    Ext           -  341
CR       Ext           -  436  1441
CRLF     Ext           - 1125
CSP      Ext           -  363
DEC      Ext           -  943  1216  1348
DOCOL    Ext           -   23    37    50    62    73    84    95   106
                          118   132   144   156   168   180   193   206
                          224   239   273   324   335   389   421   460
                          470   481   640   651   703   715   759   804
                          849   908   988  1095  1171  1357
DROP     Ext           -   39    52   312   356   398   407   624   706
                         1017  1063  1107  1177  1389
DUP      Ext           -  425   529   596   741   784   998   999  1266
EOF      Ext           -  682   719
EQUAL    Ext           -  291   596  1002
F.AND    Ext           - 1108  1180  1390
FADJ     Ext           -  826  1033
FCREAT   Ext           -  522   777
FFIND    Ext           -  279   809
FOUR+    Ext           - 1031
FROP     Ext           -  588   589
FSTX     Ext           -  442   514
FTOI     Ext           -  587
GT       Ext           -  489   871  1148  1247  1280  1296
GT#      Ext           -  215
HEX      Ext           - 1233
HOLD     Ext           -  147
IOKILL   Ext           -  355
IP       Ext           -  579
LASTX    Ext           -  580
LISTNG   Ext           -  441   445   506  1105  1175  1387
LIT      Ext           -   40   145   226   253   286   345   371   471
                          487   520   537   555   617   630   641   657
                          663   688   742   746   794   810   818   833
                          869   928  1020  1078  1190  1193  1196  1237
                         1263  1267  1294  1333  1396  1405
LT#      Ext           -  156   170   182   195   208
MAX      Ext           -  822  1200
MIN      Ext           -  748  1192  1335
MINUS    Ext           -  744   820  1255
MULT     Ext           -  228  1239  1265
N!       Ext           -  373   893
N#       Ext           -  210   211   212   213   214
NULL$    Ext           -   96   107   974  1159  1408
```



```
ONE      Ext           -  325   704   735   776   922  1068  1076  1138
                         1199  1246  1361
ONE+     Ext           -  882  1308
ONE-     Ext           -   64   225  1219  1279  1434
ONERR    Ext           -  255   262   378   557   591   665   677
OPNF     Ext           -  670
OR       Ext           -  121
OUTPUT   Ext           - 1124  1126
OVER     Ext           - 1029  1045  1049
OVER2    Ext           -   74    75
PAGESZ   Ext           -  485  1146
PRIME    Ext           - 1122
PSTOR    Ext           -  737   924  1040  1140  1363
QUOTC    Ext           -   24   243   393   402   562   570   763   936
                          958  1367  1425
R>       Ext           -  290   311
R@       Ext           -  868   881   891  1209  1236  1259  1276  1293
                         1307  1323
ROT      Ext           - 1056  1374
S!       Ext           -  109   750   772   993  1381
S<&      Ext           -  249   569   574   940   945   950   952   964
                          969  1011  1229  1258  1317  1331  1336  1436
S>&      Ext           -   97  1376  1431
SCFIB    Ext           -  339   348   655
SEMI     Ext           -   29    42    54    65    76    87    98   110
                          124   136   148   216   231   264   299   306
                          315   327   380   412   550   607   625   695
                          707   723   751   789   841   900   980  1058
                         1087  1163  1349  1416  1446
SMOVE    Ext           - 1057
STORE    Ext           -  123   256   263   349   359   366   379   499
                          502   531   549   558   592   606   659   666
                          678   786   788   921  1123  1445
SWAP     Ext           -  536  1016  1048  1106  1176  1270  1388
SWAP2    Ext           -  248   568
TWO      Ext           -   53
TWO*     Ext           - 1001  1030
TWO+     Ext           -  542  1047  1051
TWO/     Ext           - 1000
TYPE     Ext           -  437  1442
VARID    Ext           -  353   358
XDO      Ext           -  862  1202  1271
XEY?     Ext           -  581
XLOOP    Ext           -  894  1318  1342
ZBRNH    Ext           -  281   292   426   443   490   508   515   524
                          582   600   671   683   720   779   828   856
                          872   914  1004  1034  1071  1109  1117  1149
                         1181  1211  1248  1281  1297  1325  1391  1400
ZEQ      Ext           -  599   827  1003  1070  1210  1324  1399
ZERO     Ext           -  157   169   181   194   207   261   305   357
                          370   377   497   500   547   590   676   722
                          821   861   874   919  1009  1044  1201  1303
                         1443
=a&CLR   Abs  959408 #EA3B0 -  106  1162  1207
=a>OPx   Abs  959683 #EA4C3 -  224   883  1309
```

```
=a>PAD   Abs  959388 #EA39C -    95   247   567  1430
=aABRT   Abs  960176 #EA6B0 -   421   462
=aBLNK1  Abs  959328 #EA360 -    62  1286
=aBLNK2  Abs  959303 #EA347 -    50    63  1330
=aBLNK5  Abs  959273 #EA329 -    37    51  1227
=aBLNKB  Abs  959232 #EA300 -    23    38   949
=aCLEAN  Abs  960099 #EA663 -   389   451
=aCLOSE  Abs  959959 #EA5D7 -   335   411
=aDROP4  Abs  959368 #EA388 -    84
=aDUP4   Abs  959348 #EA374 -    73
=aEJECT  Abs  961397 #EAB75 -   908  1154
 aERCNT  Ext                -  1362
 aERMSG  Ext                -   435  1080  1375
 aERR$   Ext                -  1380  1414  1421
=aERROR  Abs  962690 #EB082 -  1357
 aFLSIZ  Ext                -   816
 aFLTYP  Ext                -  1244
=aKILL   Abs  959723 #EA4EB -   239   297   429
 aLC     Ext                -   785   889  1234
 aLINEC  Ext                -  1444
 aLISTF  Ext                -   501   530  1018
 aLLIN#  Ext                -   920  1139  1144
 aLLINE  Ext                -   108   935   957   992   994  1055  1208
 aLNOFF  Ext                -   548  1023  1039
=aLSTAB  Abs  960281 #EA719 -   470   556   612
 aLSTAD  Ext                -   498   605  1114
=aLSTIN  Abs  960301 #EA72D -   481
 aMEMAB  Ext                -   254
 aMEMOV  Ext                -   840
 aMONAT  Ext                -  1099
 aNEG1   Ext                -   298   314   535   597   728
=aNUM#1  Abs  959518 #EA41E -   156  1312
=aNUM#2  Abs  959543 #EA437 -   168
=aNUM#3  Abs  959568 #EA450 -   180   944
=aNUM#4  Abs  959598 #EA46E -   193  1220  1435
=aNUM#5  Abs  959628 #EA48C -   206  1257
 aOBJFL  Ext                -   787  1253
=aOBJGR  Abs  961142 #EAA76 -   804
=aOBJIN  Abs  961031 #EAA07 -   759
 aOCCUR  Ext                -   119   122   133
 aOFILE  Ext                -   461   771   808
 aOP     Ext                -   229
 aOPLEN  Ext                -   853  1188  1277
=aOPSRC  Abs  960771 #EA903 -   651
=aOUTLI  Abs  961945 #EAD99 -  1095   931   953   970   975  1341  1409  1415
=aOUTLS  Abs  962135 #EAE57 -  1171
=aOUTOB  Abs  961252 #EAAE4 -   849
 aPAGE#  Ext                -   923   941
=aPRT#   Abs  961640 #EAC68 -   988  1133
=aPURGE  Abs  959809 #EA541 -   273   326
=aRDSRC  Abs  960911 #EA98F -   715
=aREMOV  Abs  959939 #EA5C3 -   324   397   406   446
=aRSSRC  Abs  960886 #EA976 -   703
=aSETF   Abs  959433 #EA3C9 -   118  1077  1407
=aSHUTD  Abs  960266 #EA70A -   460   473   619   632   643   690   796   835
```



```
 aSLIN#  Ext                -   729   736  1217  1432
 aSLINE  Ext                -   749  1332
=aSP     Abs  959493 #EA405 -   144   183   196   209
=aSRCAB  Abs  960751 #EA8EF -   640   664
 aSTIT   Ext                -   968
=aTESTF  Abs  959468 #EA3EC -   132  1069  1398
 aTIT    Ext                -   951
 aTOLST  Ext                -   912  1103  1178  1385
 zLEFT   Abs     115 #00073 -  1449  1450
 zNEXT   Abs  963072 #EB200 -     5  1449
 zSIZE   Abs    3725 #00E8D -  1448
 zTHIS   Abs  959232 #EA300 -     4  1448
```



```
Input Parameters

  Source file name is GN&AS2

  Listing file name is GN/AS2::65

  Object file name is GN%AS2::65

                                        111111
                              0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News
```

```
 1                      TITLE  AS3:_FORTH_ASSEMBLER_SYT
 2                      RDSYMB MR%GTO
 3 EB200                ABS    #EB200
 4              zTHIS   EQU    *
 5              zNEXT   EQU    #EBB00
 6
 7              ********************************************************
 8              ***
 9              *** MR&AS3     <840504.1037>
10              ***
11              ***
12              ***  FORTH ASSEMBLER
13              *** SYMBOL TABLE
14              ***
15              ********************************************************
16              ************************************************************
17              ***
18              ***  SYT.INIT
19              ***
20              *** ( -- )
21              *** ( INITIALIZE THE SYMBOL TABLE )
22              ************************************************************
23 EB200 0000 =aSYTIN CON(5) =DOCOL         : SYT.INIT
         0
24              *
25              * CREATE SYMBOL TABLE
26              *
27 EB205 0000         CON(5) =QUOTC         "
         0
28 EB20A 60           CON(2) 6
29 EB20C 60           CON(2) 6
30 EB20E 1435         NIBASC \ASMBSY\
         D424
         3595
31 EB21A 0000         CON(5) =LIT           18          ( * STR NUM * )
         0
32 EB21F 2100         CON(5) 18
         0
33 EB224 0000         CON(5) =FCREAT        FCREATE
         0
34 EB229 0000         CON(5) =?DUP          ?DUP
         0
35 EB22E 0000         CON(5) =ZBRNH         IF          ( * [ADDR] * )
         0
36 EB233 B400         REL(5) %IF35
         0
37              *
38              * SAVE START OF SYMBOL TABLE
39              *
40 EB238 0000         CON(5) =LIT           37
         0
41 EB23D 5200         CON(5) 37
         0
42 EB242 0000         CON(5) =ADD           +           ( * ADDR * )
         0
```



```
43 EB247 0000         CON(5) =DUP           DUP
         0
44 EB24C 0000         CON(5) =aSYTAD        SYTAD
         0
45 EB251 0000         CON(5) =STORE         !           ( * ADDR * )
         0
46              *
47              * PUT IN HIGHEST POSSIBLE ENTRY INTO SYMBOL TABLE.  THIS HELPS
48              * MAKE SEARCHING & INSERTING NEW ENTRIES MUCH EASIER.
49              *
50 EB256 0000         CON(5) =LIT           9
         0
51 EB25B 9000         CON(5) 9
         0
52 EB260 0000         CON(5) =aNEG1         -1
         0
53 EB265 0000         CON(5) =FILL          FILL        ( * * )
         0
54 EB26A 0000         CON(5) =ONE           1
         0
55 EB26F 0000         CON(5) =aSYTSZ        SYTSIZE
         0
56 EB274 0000         CON(5) =STORE         !           ( * * )
         0
57 EB279 0000         CON(5) =SEMI          ;           ( * * )
         0
58              *
59              * COULDN'T CREATE SYMBOL TABLE - MUST MEAN NO MEMORY AVAILABLE
60              *
61 EB27E               %IF35                 ELSE
62 EB27E 0000         CON(5) =LIT           47
         0
63 EB283 F200         CON(5) 47             'not enought memory for assembler'
         0
64 EB288 0000         CON(5) =aSHUTD        SHUTDOWN    ( * * )
         0
65 EB28D 0000         CON(5) =SEMI          ;           ( * * )
         0
66              ************************************************************
67              ***
68              ***  >ADR
69              ***
70              *** ( POS -- PTR )
71              *** ( CALCULATE ADDRESS OF SYMBOL TABLE ENTRY POS )
72              ************************************************************
73 EB292 0000 =a>ADR  CON(5) =DOCOL         : >ADR
         0
74              *
75              * ADDR = <SYMBOL TABLE ADDRESS> + <ENTRY LENGTH> * <POSITION - 1>
76              *
77 EB297 0000         CON(5) =ONE-          1-
         0
78 EB29C 0000         CON(5) =LIT           18
         0
79 EB2A1 2100         CON(5) 18
```

```
          0
    80 EB2A6 0000         CON(5) =MULT          *
          0
    81 EB2AB 0000         CON(5) =aSYTAD        SYTAD
          0
    82 EB2B0 0000         CON(5) =AT            @
          0
    83 EB2B5 0000         CON(5) =ADD           +
          0
    84 EB2BA 0000         CON(5) =SEMI          ;           ( * PTR * )
          0
    85            ***************************************************************
    86            ***
    87            ***  ENSURE8LONG
    88            ***
    89            *** ( STR -- STR )
    90            *** ( APPEND SPACES AND ENSURE STRING EIGHT LONG )
    91            ***************************************************************
    92 EB2BF 0000 =aENS8L CON(5) =DOCOL         : ENSURE8LONG
          0
    93            *
    94            * SAVE CURRENT STRING IN ERR$ STRING VARIABLE
    95            *
    96 EB2C4 0000         CON(5) =LIT           42
          0
    97 EB2C9 A200         CON(5) 42
          0
    98 EB2CE 0000         CON(5) =MIN           MIN         ( * STR * )
          0
    99 EB2D3 0000         CON(5) =aERR$         ERR$
          0
   100 EB2D8 0000         CON(5) =S!            S!
          0
   101 EB2DD 0000         CON(5) =aERR$         ERR$        ( * STR * )
          0
   102            *
   103            * APPEND EIGHT (8) BLANKS TO THE END OF IT
   104            *
   105 EB2E2 0000         CON(5) =QUOTC         "
          0
   106 EB2E7 80           CON(2) 8
   107 EB2E9 80           CON(2) 8
   108 EB2EB 0202         NIBASC \        \
          0202
          0202
          0202
   109 EB2FB 0000         CON(5) =S<&           S<&         ( * STR * )
          0
   110            *
   111            * TAKE FIRST EIGHT (8) CHARACTERS
   112            *
   113 EB300 0000         CON(5) =DROP          DROP
          0
   114 EB305 0000         CON(5) =LIT           8
          0
```



```
   115 EB30A 8000         CON(5) 8
          0
   116 EB30F 0000         CON(5) =SEMI          ;           ( * STR * )
          0
   117            ***************************************************************
   118            ***
   119            ***  ENSURE6LONG
   120            ***
   121            *** ( STR -- STR )
   122            *** ( APPEND SPACES AND ENSURE STRING SIX LONG )
   123            ***************************************************************
   124 EB314 0000 =aENS6L CON(5) =DOCOL         : ENSURE6LONG
          0
   125 EB319 FB2B         CON(5) =aENS8L        ENSURE8LONG
          E
   126 EB31E 0000         CON(5) =DROP          DROP
          0
   127 EB323 0000         CON(5) =LIT           6
          0
   128 EB328 6000         CON(5) 6
          0
   129 EB32D 0000         CON(5) =SEMI          ;           ( * STR * )
          0
   130            ***************************************************************
   131            ***
   132            ***  DROP LEADING c
   133            ***
   134            *** ( STR c -- STR )
   135            *** ( DROP LEADING CHARACTER IF IT IS c )
   136            ***************************************************************
   137 EB332 0000 =aDROPc CON(5) =DOCOL         : DROP LEADING c
          0
   138            *
   139            * GET FIRST CHARACTER FROM STRING
   140            *
   141 EB337 0000         CON(5) =>R            >R          ( * STR * )
          0
   142 EB33C 0000         CON(5) =OVER          OVER        ( * STR ADDR * )
          0
   143 EB341 0000         CON(5) =CAT           C@          ( * STR C * )
          0
   144 EB346 0000         CON(5) =R>            R>          ( * STR C c * )
          0
   145 EB34B 0000         CON(5) =EQUAL         =           ( * STR F * )
          0
   146            *
   147            * ENSURE THERE IS A CHARACTER IN THE STRING
   148            *
   149 EB350 0000         CON(5) =OVER          OVER        ( * STR F LEN * )
          0
   150 EB355 0000         CON(5) =AND           AND         ( * STR F * )
          0
   151 EB35A 0000         CON(5) =ZBRNH         IF          ( * STR * )
          0
   152 EB35F F000         REL(5) %IF44
```

```
          0
153             *
154             * DROP FIRST CHARACTER FROM STRING
155             *
156 EB364 0000          CON(5) =C@+             C@+
          0
157 EB369 0000          CON(5) =DROP            DROP        ( * STR * )
          0
158             *
159             * ALL DONE
160             *
161 EB36E        %IF44                          THEN        ( * STR * )
162 EB36E 0000          CON(5) =SEMI            ;           ( * STR * )
          0
163             ***************************************************************
164             ***
165             ***  LOOKUP
166             ***
167             *** ( STR -- PTR )
168             *** ( LOOKUP STRING IN SYMBOL TABLE )
169             ***************************************************************
170 EB373 0000 =aLUKUP CON(5) =DOCOL          : LOOKUP
          0
171             *
172             * DROP LEADING = SIGN
173             *
174 EB378 0000          CON(5) =LIT             \=\         ( * STR NUM * )
          0
175 EB37D D300          CON(5) \=\
          0
176 EB382 233B          CON(5) =aDROPc          DROPc       ( * STR * )
          E
177             *
178             * INITIALIZE VARIABLES
179             *   STR - MAX SIX CHARACTERS LONG
180             *   BFOUND - FALSE, NOT FOUND.
181             *   BLOW - ONE (1), FIRST ENTRY.
182             *   BTOP - <SYMBOL TABLE SIZE>, LAST ENTRY.
183             *
184 EB387 413B          CON(5) =aENS6L          ENSURE6LONG ( * STR * )
          E
185 EB38C 0000          CON(5) =ZERO            0
          0
186 EB391 0000          CON(5) =aBFOUN          BFOUND
          0
187 EB396 0000          CON(5) =STORE           !           ( * STR * )
          0
188 EB39B 0000          CON(5) =ONE             1
          0
189 EB3A0 0000          CON(5) =aBLOW           BLOW
          0
190 EB3A5 0000          CON(5) =STORE           !           ( * STR * )
          0
191 EB3AA 0000          CON(5) =aSYTSZ          SYTSIZE
          0
```



```
192 EB3AF 0000          CON(5) =AT              @
          0
193 EB3B4 0000          CON(5) =aBTOP           BTOP
          0
194 EB3B9 0000          CON(5) =STORE           !           ( * STR * )
          0
195             *
196             * while BLOW <= BTOP and NOT BFOUND do
197             *
198 EB3BE        %BG1                           BEGIN       ( * STR * )
199 EB3BE 0000          CON(5) =aBLOW           BLOW
          0
200 EB3C3 0000          CON(5) =AT              @
          0
201 EB3C8 0000          CON(5) =aBTOP           BTOP
          0
202 EB3CD 0000          CON(5) =AT              @
          0
203 EB3D2 0000          CON(5) =GT              >           ( * [STR] F * )
          0
204 EB3D7 0000          CON(5) =aBFOUN          BFOUND
          0
205 EB3DC 0000          CON(5) =AT              @
          0
206 EB3E1 0000          CON(5) =OR              OR
          0
207 EB3E6 0000          CON(5) =ZEQ             0=          ( * [STR] F * )
          0
208 EB3EB 0000          CON(5) =ZBRNH           WHILE       ( * STR * )
          0
209 EB3F0 DC00          REL(5) %RP1
          0
210             *
211             * CALCULATE MIDDLE ENTRY ((BTOP + BLOW) / 2)
212             *
213 EB3F5 0000          CON(5) =aBTOP           BTOP
          0
214 EB3FA 0000          CON(5) =AT              @
          0
215 EB3FF 0000          CON(5) =aBLOW           BLOW
          0
216 EB404 0000          CON(5) =AT              @
          0
217 EB409 0000          CON(5) =ADD             +
          0
218 EB40E 0000          CON(5) =TWO/            2/          ( * STR MID * )
          0
219             *
220             * SAVE LAST ENTRY LOOKED AT IN VARIABLE SYTLAST.  THIS
221             * MARKS WHERE TO INSERT THE ENTRY IF IT ISN'T FOUND.
222             *
223 EB413 0000          CON(5) =DUP             DUP
          0
224 EB418 0000          CON(5) =aSYTLS          SYTLAST
          0
```

```
225 EB41D 0000          CON(5) =STORE           !
          0
226             *
227             * EXTRACT STRING FROM THAT ENTRY
228             *
229 EB422 292B          CON(5) =a>ADR           >ADR
          E
230 EB427 0000          CON(5) =LIT             6           ( * STR STR' * )
          0
231 EB42C 6000          CON(5) 6
          0
232 EB431 0000          CON(5) =aDUP4           4DUP
          0
233 EB436 0000          CON(5) =S=              S=
          0
234 EB43B 0000          CON(5) =ZBRNH           IF          ( * STR STR' * )
          0
235 EB440 3200          REL(5) %IF36
          0
236             *
237             * FOUND IT!  SET VARIABLE BFOUND
238             *
239 EB445 0000          CON(5) =aDROP4          4DROP       ( * * )
          0
240 EB44A 0000          CON(5) =aNEG1           -1
          0
241 EB44F 0000          CON(5) =aBFOUN          BFOUND
          0
242 EB454 0000          CON(5) =STORE           !           ( * * )
          0
243 EB459 0000          CON(5) =BRNCH           ELSE        ( * STR STR' * )
          0
244 EB45E 5500          REL(5) %TH36
          0
245             *
246             * STRINGS NOT THE SAME.  DO WE GO UP OR DOWN?
247             *
248 EB463        %IF36
249 EB463 0000          CON(5) =OVER2           2OVER
          0
250 EB468 0000          CON(5) =S<              S<
          0
251 EB46D 0000          CON(5) =ZBRNH           IF          ( * STR * )
          0
252 EB472 8200          REL(5) %IF37
          0
253             *
254             * WE GO UP
255             *
256 EB477 0000          CON(5) =aSYTLS          SYTLAST
          0
257 EB47C 0000          CON(5) =AT              @           ( * STR POS * )
          0
258 EB481 0000          CON(5) =ONE+            1+
          0
```



```
259 EB486 0000          CON(5) =aBLOW           BLOW
          0
260 EB48B 0000          CON(5) =STORE           !           ( * STR * )
          0
261 EB490 0000          CON(5) =BRNCH           ELSE        ( * STR * )
          0
262 EB495 E100          REL(5) %TH37
          0
263             *
264             * WE GO DOWN
265             *
266 EB49A        %IF37
267 EB49A 0000          CON(5) =aSYTLS          SYTLAST
          0
268 EB49F 0000          CON(5) =AT              @           ( * STR POS * )
          0
269 EB4A4 0000          CON(5) =ONE-            1-
          0
270 EB4A9 0000          CON(5) =aBTOP           BTOP
          0
271 EB4AE 0000          CON(5) =STORE           !           ( * STR * )
          0
272 EB4B3        %TH37                          THEN        ( * STR * )
273             *
274             * end while
275             *
276 EB4B3        %TH36                          THEN        ( * [STR] * )
277 EB4B3 0000          CON(5) =BRNCH           REPEAT      ( * [STR] * )
          0
278 EB4B8 60FF          REL(5) %BG1
          F
279             *
280             * IF FOUND, RETURN ADDRESS OF ENTRY
281             *
282 EB4BD        %RP1
283 EB4BD 0000          CON(5) =aBFOUN          BFOUND
          0
284 EB4C2 0000          CON(5) =AT              @
          0
285 EB4C7 0000          CON(5) =ZBRNH           IF          ( * [STR] * )
          0
286 EB4CC 9100          REL(5) %IF38
          0
287 EB4D1 0000          CON(5) =aSYTLS          SYTLAST
          0
288 EB4D6 0000          CON(5) =AT              @
          0
289 EB4DB 292B          CON(5) =a>ADR           >ADR        ( * PTR * )
          E
290 EB4E0 0000          CON(5) =SEMI            ;           ( * PTR * )
          0
291             *
292             * IF NOT FOUND, SAVE ADDRESS OF PLACE TO INSERT AND RETURN FF
293             *
294 EB4E5        %IF38                          ELSE        ( * STR * )
```

```
295 EB4E5 0000          CON(5) =aSYTLS          SYTLAST
          0
296 EB4EA 0000          CON(5) =AT              @
          0
297 EB4EF 0000          CON(5) =aBLOW           BLOW
          0
298 EB4F4 0000          CON(5) =AT              @           ( * STR LST LOW * )
          0
299 EB4F9 0000          CON(5) =MAX             MAX
          0
300 EB4FE 0000          CON(5) =aSYTLS          SYTLAST
          0
301 EB503 0000          CON(5) =STORE           !           ( * STR * )
          0
302 EB508 0000          CON(5) =2DROP           2DROP
          0
303 EB50D 0000          CON(5) =ZERO            0           ( * PTR * )
          0
304 EB512 0000          CON(5) =SEMI            ;           ( * PTR * )
          0
305             ****************************************************************
306             ***
307             ***  SYT>
308             ***
309             *** ( STR -- NUM )
310             *** ( RETRIEVE A VALUE FROM THE SYMBOL TABLE )
311             *** ( IF FOUND, THEN VARIABLE KNOWN IS SET TRUE )
312             ****************************************************************
313 EB517 0000 =aSYT>  CON(5) =DOCOL           : SYT>
          0
314             *
315             * LOOK UP VALUE FIRST
316             *
317 EB51C 373B            CON(5) =aLUKUP          LOOKUP
          E
318 EB521 0000          CON(5) =?DUP            ?DUP
          0
319 EB526 0000          CON(5) =ZBRNH           IF          ( * [PTR] * )
          0
320 EB52B F500          REL(5) %IF39
          0
321             *
322             * FOUND, RETURN VALUE
323             *
324 EB530 0000          CON(5) =LIT             12
          0
325 EB535 C000          CON(5) 12
          0
326 EB53A 0000          CON(5) =ADD             +           ( * PTR * )
          0
327 EB53F 0000          CON(5) =DUP             DUP         ( * PTR PTR * )
          0
328 EB544 0000          CON(5) =AT              @           ( * PTR NUM * )
          0
329             *
```



```
330             * LOOK AT TYPE NIBBLE:
331             *   1 = LABEL
332             *   2 = EQUATE
333             *
334 EB549 0000          CON(5) =SWAP            SWAP        ( * NUM PTR * )
          0
335 EB54E 0000          CON(5) =FIVE+           5+          ( * NUM PTR * )
          0
336 EB553 0000          CON(5) =N@              N@          ( * NUM NIB * )
          0
337 EB558 0000          CON(5) =ONE             1
          0
338 EB55D 0000          CON(5) =AND             AND         ( * NUM BIT0 * )
          0
339 EB562 0000          CON(5) =ZBRNH           IF
          0
340 EB567 4100          REL(5) %IF313
          0
341 EB56C 0000          CON(5) =aNEG1           -1          ( * NUM -1 * ) LABEL
          0
342 EB571 0000          CON(5) =BRNCH           ELSE
          0
343 EB576 A000          REL(5) %TH313
          0
344 EB57B       %IF313
345 EB57B 0000          CON(5) =ONE             1           ( * NUM 1 * )  EQUATE
          0
346 EB580       %TH313                          THEN
347 EB580 0000          CON(5) =BRNCH           ELSE
          0
348 EB585 F000          REL(5) %TH39
          0
349             *
350             * NOT FOUND, RETURN 0 AND FALSE IN VARIABLE KNOWN
351             *
352 EB58A       %IF39
353 EB58A 0000          CON(5) =ZERO            0
          0
354 EB58F 0000          CON(5) =ZERO            0           ( * NUM 0 * )  NOT FOUN
          0
355             *
356             * SET VARIABLE KNOWN:
357             *   0 = NOT KNOWN
358             *   1 = EQUATE
359             *  -1 = LABEL
360             *
361 EB594       %TH39                           THEN        ( * NUM TYP * )
362 EB594 0000          CON(5) =aKNOWN          KNOWN       ( * NUM TYP ADDR * )
          0
363 EB599 0000          CON(5) =STORE           !           ( * NUM * )
          0
364 EB59E 0000          CON(5) =SEMI            ;           ( * NUM * )
          0
365             ****************************************************************
366             ***
```

```
367             ***  LABEL.OK?
368             ***
369             *** ( STR -- F )
370             *** ( F TRUE MEANS THE LABEL IS A VALID LABEL )
371             ********************************************************************
372 EB5A3 0000 =aLABOK CON(5) =DOCOL          : LABEL.OK?
        0
373             *
374             * DROP LEADING =
375             *
376 EB5A8 0000         CON(5) =LIT           61          ( * STR NUM * )
        0
377 EB5AD D300         CON(5) \=\
        0
378 EB5B2 233B         CON(5) =aDROPc        DROPc       ( * STR * )
        E
379             *
380             * EXTRACT FIRST CHARACTER
381             *
382 EB5B7 0000         CON(5) =OVER          OVER        ( * STR ADDR * )
        0
383 EB5BC 0000         CON(5) =CAT           C@          ( * STR C * )
        0
384             *
385             * FIRST CHARACTER A '#' ?
386             *
387 EB5C1 0000         CON(5) =DUP           DUP         ( * STR C C * )
        0
388 EB5C6 0000         CON(5) =LIT           35          ( * STR C NUM * )
        0
389 EB5CB 3200         CON(5) \#\
        0
390 EB5D0 0000         CON(5) =EQUAL         =           ( * STR C F * )
        0
391             *
392             * FIRST CHARACTER A ''' ?
393             *
394 EB5D5 0000         CON(5) =OVER          OVER        ( * STR C F C * )
        0
395 EB5DA 0000         CON(5) =LIT           39          ( * STR C F C NUM * )
        0
396 EB5DF 7200         CON(5) \'\
        0
397 EB5E4 0000         CON(5) =EQUAL         =           ( * STR C F F' * )
        0
398 EB5E9 0000         CON(5) =OR            OR          ( * STR C F * )
        0
399             *
400             * FIRST CHARACTER A '('
401             *
402 EB5EE 0000         CON(5) =OVER          OVER        ( * STR C F C * )
        0
403 EB5F3 0000         CON(5) =LIT           40          ( * STR C F C NUM * )
        0
404 EB5F8 8200         CON(5) \(\
```



```
        0
405 EB5FD 0000         CON(5) =EQUAL         =           ( * STR C F F' * )
        0
406 EB602 0000         CON(5) =OR            OR          ( * STR C F * )
        0
407             *
408             * FIRST CHARACTER A DECIMAL DIGIT ?
409             *
410 EB607 0000         CON(5) =OVER          OVER        ( * STR C F C * )
        0
411 EB60C 0000         CON(5) =aDIG10        DIGIT.10    ( * STR C F F * )
        0
412 EB611 0000         CON(5) =OR            OR          ( * STR C F * )
        0
413             *
414             * FIRST CHARACTER A '-' ?
415             *
416 EB616 0000         CON(5) =SWAP          SWAP        ( * STR F C * )
        0
417 EB61B 0000         CON(5) =LIT           45          ( * STR F C NUM * )
        0
418 EB620 D200         CON(5) \-\
        0
419 EB625 0000         CON(5) =EQUAL         =           ( * STR F F * )
        0
420 EB62A 0000         CON(5) =OR            OR          ( * STR F * )
        0
421             *
422             * BURY THIS FLAG, IT'S USED IN THE LOOP
423             *
424 EB62F 0000         CON(5) =ROT           ROT
        0
425 EB634 0000         CON(5) =ROT           ROT         ( * F STR * )
        0
426 EB639 0000         CON(5) =ZERO          0           ( * F ADDR LEN 0 * )
        0
427             *
428             * LOOP THROUGH EACH CHARACTER
429             *
430 EB63E 0000         CON(5) =XDO           DO          ( * F ADDR * )
        0
431 EB643       %DO20
432             *
433             * EXTRACT CHARACTER
434             *
435 EB643 0000         CON(5) =DUP           DUP         ( * F ADDR ADDR * )
        0
436 EB648 0000         CON(5) =CAT           C@          ( * F ADDR C * )
        0
437             *
438             * IS THE CHARACTER A ',' ?
439             *
440 EB64D 0000         CON(5) =DUP           DUP         ( * F ADDR C C * )
        0
441 EB652 0000         CON(5) =LIT           44          ( * F ADDR C C NUM * )
```

```
          0
442 EB657 C200          CON(5) \,\
          0
443 EB65C 0000          CON(5) =EQUAL           =           ( * F ADDR C F' * )
          0
444               *
445               * IS THE CHARACTER A ')' ?
446               *
447 EB661 0000          CON(5) =SWAP            SWAP        ( * F ADDR F' C * )
          0
448 EB666 0000          CON(5) =LIT             41          ( * F ADDR F' C NUM * )
          0
449 EB66B 9200          CON(5) \)\
          0
450 EB670 0000          CON(5) =EQUAL           =           ( * F ADDR F' F" * )
          0
451 EB675 0000          CON(5) =OR              OR          ( * F ADDR F' * )
          0
452 EB67A 0000          CON(5) =ROT             ROT         ( * ADDR F' F * )
          0
453 EB67F 0000          CON(5) =OR              OR          ( * ADDR F * )
          0
454               *
455               * BUMP ADDRESS TO NEXT CHARACTER
456               *
457 EB684 0000          CON(5) =SWAP            SWAP        ( * F ADDR * )
          0
458 EB689 0000          CON(5) =TWO+            2+          ( * F ADDR * )
          0
459 EB68E 0000          CON(5) =XLOOP           LOOP        ( * F ADDR * )
          0
460 EB693 0BFF          REL(5) %D020
          F
461               *
462               * RETURN FALSE IF ANY OF ABOVE QUESTIONS TRUE
463               *
464 EB698 0000          CON(5) =DROP            DROP        ( * F * )
          0
465 EB69D 0000          CON(5) =ZEQ             0=          ( * F * )
          0
466 EB6A2 0000          CON(5) =SEMI            ;           ( * F * )
          0
467               ****************************************************************
468               ***
469               ***  >SYT
470               ***
471               *** ( NUM STR -- )
472               *** ( PUT A VALUE FOR A LABEL IN THE SYMBOL TABLE )
473               ****************************************************************
474 EB6A7 0000 =a>SYT  CON(5) =DOCOL           : >SYT
          0
475               *
476               * VALID LABEL?
477               *
478 EB6AC 0000          CON(5) =2DUP            2DUP        ( * NUM STR STR * )
```



```
          0
479 EB6B1 3A5B          CON(5) =aLABOK          LABEL.OK?   ( * NUM STR F * )
          E
480 EB6B6 0000          CON(5) =ZBRNH           IF          ( * NUM STR * )
          0
481 EB6BB 7D00          REL(5) %IF309
          0
482               *
483               * YES.  DOES IT ALREADY EXIST?
484               *
485 EB6C0 373B          CON(5) =aLUKUP          LOOKUP
          E
486 EB6C5 0000          CON(5) =ZBRNH           IF          ( * NUM * )
          0
487 EB6CA F000          REL(5) %IF40
          0
488               *
489               * YES.  THIS ISN'T THE ROUTINE TO REPORT THE ERROR.  IF FIRST PAS
490               * ERROR SHOULD BE ISSUED.  DUPLICATE LABEL ERRORS ARE GENERATED B
491               * THE VALUE FOR THE LABEL IS DIFFERENT THAN THE ONE IN THE SYMBOL
492               * NOT BECAUSE OF AN EXPLICIT CHECK.
493               *
494 EB6CF 0000          CON(5) =DROP            DROP        ( * * )
          0
495 EB6D4 0000          CON(5) =SEMI            ;           ( * * )
          0
496               *
497               * NEW LABEL
498               *
499 EB6D9                %IF40                  ELSE        ( * NUM * )
500 EB6D9 0000          CON(5) =ONE             1
          0
501 EB6DE 0000          CON(5) =aSYTSZ          SYTSIZE
          0
502 EB6E3 0000          CON(5) =PSTOR           +!          ( * NUM * )
          0
503               *
504               * GET PLACE TO INSERT LABEL
505               *
506 EB6E8 0000          CON(5) =aSYTLS          SYTLAST
          0
507 EB6ED 0000          CON(5) =AT              @
          0
508 EB6F2 292B          CON(5) =a>ADR           >ADR        ( * NUM PTR * )
          E
509               *
510               * ADJUST SYMBOL TABLE TO INCLUDE LABEL
511               *
512 EB6F7 0000          CON(5) =DUP             DUP
          0
513 EB6FC 0000          CON(5) =LIT             18
          0
514 EB701 2100          CON(5) 18
          0
515 EB706 0000          CON(5) =FADJ            FADJUST
```

```
          0
   516 EB70B 0000        CON(5) =ZBRNH         IF          ( * NUM PTR * )
          0
   517 EB710 3700        REL(5) %IF41
          0
   518             *
   519             * SUCCESS.  LOOKUP PUT THE LABEL INTO ERR$ (ACTUALLY, ENSURE6LONG
   520             * DID, BUT WHO'S LOOKING?)  MOVE THE STRING INTO THE NEW ENTRY
   521             *
   522 EB715 0000        CON(5) =DUP           DUP
          0
   523 EB71A 0000        CON(5) =>R            >R          ( * NUM PTR * )
          0
   524 EB71F 0000        CON(5) =aERR$         ERR$
          0
   525 EB724 0000        CON(5) =DROP          DROP        ( * NUM PTR ADDR * )
          0
   526 EB729 0000        CON(5) =SWAP          SWAP
          0
   527 EB72E 0000        CON(5) =LIT           6
          0
   528 EB733 6000        CON(5) 6
          0
   529 EB738 0000        CON(5) =CMOVE         CMOVE
          0
   530 EB73D 0000        CON(5) =R>            R>          ( * NUM PTR * )
          0
   531             *
   532             * ASSUME FOR NOW, THAT THIS IS A LABEL:  STORE 1 IN TYPE NIBBLE.
   533             *
   534 EB742 0000        CON(5) =DUP           DUP         ( * NUM PTR PTR * )
          0
   535 EB747 0000        CON(5) =LIT           17
          0
   536 EB74C 1100        CON(5) 17
          0
   537 EB751 0000        CON(5) =ADD           +           ( * NUM PTR PTR' * )
          0
   538 EB756 0000        CON(5) =ONE           1
          0
   539 EB75B 0000        CON(5) =SWAP          SWAP
          0
   540 EB760 0000        CON(5) =N!            N!          ( * NUM PTR * )
          0
   541             *
   542             * STORE VALUE FOR LABEL INTO NEW ENTRY.
   543             *
   544 EB765 0000        CON(5) =LIT           12
          0
   545 EB76A C000        CON(5) 12
          0
   546 EB76F 0000        CON(5) =ADD           +
          0
   547 EB774 0000        CON(5) =STORE         !           ( * * )
          0
```



```
   548             *
   549             * MEMORY MOVED, THE SYMBOL TABLE GREW.
   550             *
   551 EB779 0000        CON(5) =aMEMOV        MEMMOVE     ( * * )
          0
   552 EB77E 0000        CON(5) =SEMI          ;           ( * * )
          0
   553             *
   554             * FADJUST FAILED.  NO ROOM.
   555             *
   556 EB783              %IF41                 ELSE
   557 EB783 0000        CON(5) =LIT           42
          0
   558 EB788 A200        CON(5) 42             'symbol table full'
          0
   559 EB78D 0000        CON(5) =aSHUTD        SHUTDOWN    ( * * )
          0
   560             *
   561             * INVALID LABEL.  IGNORE FOR NOW.  SEE NOTE ABOVE ON DUPLICATE LA
   562             *
   563 EB792              %IF309                ELSE        ( * NUM STR * )
   564 EB792 0000        CON(5) =2DROP         2DROP       ( * NUM * )
          0
   565 EB797 0000        CON(5) =DROP          DROP        ( * * )
          0
   566 EB79C 0000        CON(5) =SEMI          ;           ( * * )
          0
   567             ****************************************************************
   568             ***
   569             ***  OUT.SYT
   570             ***
   571             *** ( -- )
   572             *** ( PRINT OUT THE SYMBOL TABLE )
   573             ****************************************************************
   574 EB7A1 0000 =aOUTSY CON(5) =DOCOL        : OUT.SYT
          0
   575             *
   576             * NEW SUBTITLE FOR SYMBOL TABLE PAGE
   577             *
   578 EB7A6 0000        CON(5) =QUOTC         "
          0
   579 EB7AB 61          CON(2) 22
   580 EB7AD 61          CON(2) 22
   581 EB7AF A2A2        NIBASC \**** SYM\
          A2A2
          0235
          95D4
   582 EB7BF 24F4        NIBASC \BOL TABL\
          C402
          4514
          24C4
   583 EB7CF 5402        NIBASC \E ****\
          A2A2
          A2A2
   584 EB7DB 0000        CON(5) =aSTIT         STIT
```

```
          0
585 EB7E0 0000        CON(5) =S!              S!
          0
586               *
587               * START NEW PAGE
588               *
589 EB7E5 0000        CON(5) =aEJECT          EJECT
          0
590 EB7EA 0000        CON(5) =a&CLR           &CLEAR
          0
591 EB7EF 0000        CON(5) =aSYTAD          SYTAD
          0
592 EB7F4 0000        CON(5) =AT              @
          0
593 EB7F9 0000        CON(5) =HEX             HEX         ( * PTR * )
          0
594               *
595               * while <SYMBOL TABLE ENTRIES> do
596               *
597 EB7FE              %BG2                   BEGIN       ( * PTR * )
598 EB7FE 0000        CON(5) =DUP             DUP
          0
599 EB803 0000        CON(5) =AT              @
          0
600 EB808 0000        CON(5) =aNEG1           -1
          0
601 EB80D 0000        CON(5) =EQUAL           =
          0
602 EB812 0000        CON(5) =ZEQ             0=          ( * PTR F * )
          0
603 EB817 0000        CON(5) =ZBRNH           WHILE       ( * PTR * )
          0
604 EB81C 7D00        REL(5) %RP2
          0
605               *
606               * EXTRACT LABEL FROM ENTRY
607               *
608 EB821 0000        CON(5) =DUP             DUP
          0
609 EB826 0000        CON(5) =LIT             6           ( * PTR STR * )
          0
610 EB82B 6000        CON(5) 6
          0
611               *
612               * APPEND TO LISTING LINE
613               *
614 EB830 0000        CON(5) =aLLINE          LLINE
          0
615 EB835 0000        CON(5) =SWAP2           2SWAP
          0
616 EB83A 0000        CON(5) =S<&             S<&         ( * PTR STR * )
          0
617 EB83F 0000        CON(5) =aBLNK2          2BLANK
          0
618 EB844 0000        CON(5) =S<&             S<&         ( * PTR STR * )
```



```
          0
619               *
620               * EXTRACT VALUE FROM ENTRY
621               *
622 EB849 0000        CON(5) =ROT             ROT
          0
623 EB84E 0000        CON(5) =LIT             12
          0
624 EB853 C000        CON(5) 12
          0
625 EB858 0000        CON(5) =ADD             +
          0
626 EB85D 0000        CON(5) =DUP             DUP
          0
627 EB862 0000        CON(5) =AT              @           ( * STR PTR NUM * )
          0
628 EB867 0000        CON(5) =OVER            OVER        ( * STR PTR NUM PTR * )
          0
629               *
630               * CHECK IF A LABEL AND WE'RE NOT A FORTH FILE (LEX OR BIN)
631               *
632 EB86C 0000        CON(5) =FIVE+           5+          ( * STR PTR NUM PTR' *
          0
633 EB871 0000        CON(5) =N@              N@          ( * STR PTR NUM F * )
          0
634 EB876 0000        CON(5) =ONE             1
          0
635 EB87B 0000        CON(5) =AND             AND         ( * STR PTR NUM F * )
          0
636 EB880 0000        CON(5) =aFLTYP          FILETYPE
          0
637 EB885 0000        CON(5) =AT              @           ( * STR PTR NUM F TYP *
          0
638 EB88A 0000        CON(5) =ONE             1
          0
639 EB88F 0000        CON(5) =GT              >
          0
640 EB894 0000        CON(5) =AND             AND         ( * STR PTR NUM F * )
          0
641 EB899 0000        CON(5) =ZBRNH           IF          ( * STR PTR NUM * )
          0
642 EB89E 4100        REL(5) %IF310
          0
643               *
644               * WE ARE:  THEN VALUE IN SYMBOL TABLE SHOULD BE RELATIVE TO START
645               * OF FILE.
646               *
647 EB8A3 0000        CON(5) =aOBJFL          OBJFILE     ( * STR PTR NUM ADDR *
          0
648 EB8A8 0000        CON(5) =AT              @
          0
649 EB8AD 0000        CON(5) =MINUS           -           ( * STR PTR NUM * )
          0
650 EB8B2              %IF310                 THEN        ( * STR PTR NUM * )
651               *
```

```
                * APPEND VALUE TO LISTING LINE
                *
EB8B2 0000      CON(5) =aNUM#5      #5          ( * STR PTR STR' * )
      0
EB8B7 0000      CON(5) =ROT         ROT
      0
EB8BC 0000      CON(5) =>R          >R          ( * STR STR' * )
      0
EB8C1 0000      CON(5) =S<&         S<&         ( * STR * )
      0
EB8C6 0000      CON(5) =R>          R>
      0
                *
                * OUTPUT THIS LINE
                *
EB8CB 0000      CON(5) =ROT         ROT
      0
EB8D0 0000      CON(5) =ROT         ROT         ( * PTR STR * )
      0
EB8D5 0000      CON(5) =aOUTLI      OUT.LINE
      0
                *
                * NEXT ENTRY
                *
EB8DA 0000      CON(5) =LIT         6
      0
EB8DF 6000      CON(5) 6
      0
EB8E4 0000      CON(5) =ADD         +           ( * PTR * )
      0
EB8E9 0000      CON(5) =BRNCH       REPEAT      ( * PTR * )
      0
EB8EE 01FF      REL(5) %BG2
      F
                *
                * end while
                *
EB8F3           %RP2
EB8F3 0000      CON(5) =DROP        DROP        ( * * )
      0
                *
                * START ASSEMBLY STATISTICS
                *
EB8F8 0000      CON(5) =NULL$       NULL$       ( * STR * )
      0
EB8FD 0000      CON(5) =aSTIT       STITLE
      0
EB902 0000      CON(5) =S!          S!          ( * * )
      0
EB907 0000      CON(5) =DEC         DECIMAL     ( * * )
      0
                *
                * SIX (6) BLANK LINES
                *
EB90C 0000      CON(5) =LIT         6
```



```
      0
EB911 6000      CON(5) 6
      0
EB916 0000      CON(5) =ZERO        0
      0
EB91B 0000      CON(5) =XDO         DO          ( * * )
      0
EB920       %DO5
EB920 0000      CON(5) =NULL$       NULL$
      0
EB925 0000      CON(5) =aOUTLI      OUT.LINE    ( * * )
      0
EB92A 0000      CON(5) =XLOOP       LOOP        ( * * )
      0
EB92F 1FFF      REL(5) %DO5
      F
                *
                * REPORT SOURCE FILE NAME
                *
EB934 0000      CON(5) =QUOTC       "  SOURCE : "
      0
EB939 A0        CON(2) 10
EB93B A0        CON(2) 10
EB93D 0235      NIBASC \ SOURCE \
      F455
      2534
      5402
EB94D A302      NIBASC \: \
EB951 0000      CON(5) =a>PAD       >PAD        ( * STR * )
      0
EB956 0000      CON(5) =aSFILE      SFILE
      0
EB95B 0000      CON(5) =S<&         S<&
      0
EB960 0000      CON(5) =aOUTLI      OUT.LINE    ( * * )
      0
EB965 0000      CON(5) =NULL$       NULL$
      0
EB96A 0000      CON(5) =aOUTLI      OUT.LINE    ( * * )
      0
                *
                * REPORT OBJECT FILE NAME
                *
EB96F 0000      CON(5) =QUOTC       "  OBJECT : "
      0
EB974 A0        CON(2) 10
EB976 A0        CON(2) 10
EB978 02F4      NIBASC \ OBJECT \
      24A4
      5434
      4502
EB988 A302      NIBASC \: \
EB98C 0000      CON(5) =a>PAD       >PAD        ( * STR * )
      0
EB991 0000      CON(5) =aOFILE      OFILE
```

```
          0
721 EB996 0000          CON(5) =?DUP            ?DUP
          0
722 EB99B 0000          CON(5) =ZEQ             0=
          0
723 EB9A0 0000          CON(5) =ZBRNH           IF          ( * STR ADDR [LEN] * )
          0
724 EB9A5 3200          REL(5) %IF43
          0
725             *
726             * SUBSTITUTE FORTHRAM WHEN NOT CREATING LEX OR BIN
727             *
728 EB9AA 0000          CON(5) =DROP            DROP
          0
729 EB9AF 0000          CON(5) =QUOTC           " FORTHRAM"
          0
730 EB9B4 80            CON(2) 8
731 EB9B6 80            CON(2) 8
732 EB9B8 64F4          NIBASC \FORTHRAM\
          2545
          8425
          14D4
733 EB9C8       %IF43                           THEN        ( * STR STR' * )
734 EB9C8 0000          CON(5) =S<&             S<&
          0
735 EB9CD 0000          CON(5) =aOUTLI          OUT.LINE    ( * * )
          0
736 EB9D2 0000          CON(5) =NULL$           NULL$
          0
737 EB9D7 0000          CON(5) =aOUTLI          OUT.LINE    ( * * )
          0
738             *
739             * REPORT LISTING FILE/DEVICE
740             *
741 EB9DC 0000          CON(5) =QUOTC           " LISTING : "
          0
742 EB9E1 A0            CON(2) 10
743 EB9E3 A0            CON(2) 10
744 EB9E5 C494          NIBASC \LISTING \
          3545
          94E4
          7402
745 EB9F5 A302          NIBASC \: \
746 EB9F9 0000          CON(5) =a>PAD           >PAD        ( * STR * )
          0
747 EB9FE 0000          CON(5) =LISTNG          LISTING
          0
748 EBA03 0000          CON(5) =S<&             S<&
          0
749 EBA08 0000          CON(5) =aOUTLI          OUT.LINE    ( * * )
          0
750 EBA0D 0000          CON(5) =NULL$           NULL$
          0
751 EBA12 0000          CON(5) =aOUTLI          OUT.LINE
          0
```



```
752 EBA17 0000          CON(5) =aLLINE          LLINE       ( * STR * )
          0
753             *
754             * REPORT TIME AND DATE
755             *
756 EBA1C 0000          CON(5) =QUOTC           "    DATE : "
          0
757 EBA21 A0            CON(2) 10
758 EBA23 A0            CON(2) 10
759 EBA25 0202          NIBASC \   DATE \
          0244
          1445
          5402
760 EBA35 A302          NIBASC \: \
761 EBA39 0000          CON(5) =S<&             S<&         ( * STR * )
          0
762 EBA3E 0000          CON(5) =QUOTC           " TIME$"
          0
763 EBA43 50            CON(2) 5
764 EBA45 50            CON(2) 5
765 EBA47 4594          NIBASC \TIME$\
          D454
          42
766 EBA51 0000          CON(5) =aBAS$           BASIC$
          0
767 EBA56 0000          CON(5) =S<&             S<&         ( * STR * )
          0
768 EBA5B 0000          CON(5) =QUOTC           "  on "
          0
769 EBA60 40            CON(2) 4
770 EBA62 40            CON(2) 4
771 EBA64 02F6          NIBASC \ on \
          E602
772 EBA6C 0000          CON(5) =S<&             S<&         ( * STR * )
          0
773 EBA71 0000          CON(5) =QUOTC           " DATE$"
          0
774 EBA76 50            CON(2) 5
775 EBA78 50            CON(2) 5
776 EBA7A 4414          NIBASC \DATE$\
          4554
          42
777 EBA84 0000          CON(5) =aBAS$           BASIC$
          0
778 EBA89 0000          CON(5) =S<&             S<&         ( * STR * )
          0
779 EBA8E 0000          CON(5) =aOUTLI          OUT.LINE    ( * * )
          0
780 EBA93 0000          CON(5) =NULL$           NULL$
          0
781 EBA98 0000          CON(5) =aOUTLI          OUT.LINE
          0
782             *
783             * REPORT NUMBER OF ERRORS
784             *
```

```
785 EBA9D 0000          CON(5) =QUOTC          "  ERRORS : "
          0
786 EBAA2 A0            CON(2) 10
787 EBAA4 A0            CON(2) 10
788 EBAA6 0254          NIBASC \ ERRORS \
          2525
          F425
          3502
789 EBAB6 A302          NIBASC \: \
790 EBABA 0000          CON(5) =a>PAD          >PAD
          0
791 EBABF 0000          CON(5) =aERCNT         ERROR COUNT
          0
792 EBAC4 0000          CON(5) =AT             @
          0
793 EBAC9 0000          CON(5) =aNUM#3         #3
          0
794 EBACE 0000          CON(5) =S<&            S<&
          0
795 EBAD3 0000          CON(5) =aOUTLI         OUT.LINE
          0
796             *
797             * DO A PAGE EJECT.  SUBTRACT ONE FROM VARIABLE LLINE# SO THAT
798             * THE NEW LINE WON'T CAUSE A PAGE EJECT ITSELF.
799             *
800 EBAD8 0000          CON(5) =aNEG1          -1
          0
801 EBADD 0000          CON(5) =aLLIN#         LLINE#
          0
802 EBAE2 0000          CON(5) =PSTOR          +!          ( * * )
          0
803 EBAE7 0000          CON(5) =LIT            12
          0
804 EBAEC C000          CON(5) 12
          0
805 EBAF1 0000          CON(5) =CHR$           CHR$
          0
806 EBAF6 0000          CON(5) =aOUTLI         OUT.LINE
          0
807 EBAFB 0000          CON(5) =SEMI           ;           ( * * )
          0
```



```
808                     STITLE FIT STATS
809             zSIZE   EQU    (*)-zTHIS
810             zLEFT   EQU    (zNEXT)-*
811 EBB00               BSS    zLEFT
812 EBB00               END
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER

%BG1	Abs	963518	#EB3BE	-	198	278					
%BG2	Abs	964606	#EB7FE	-	597	672					
%D020	Abs	964163	#EB643	-	431	460					
%D05	Abs	964896	#EB920	-	692	696					
%IF309	Abs	964498	#EB792	-	563	481					
%IF310	Abs	964786	#EB8B2	-	650	642					
%IF313	Abs	963963	#EB57B	-	344	340					
%IF35	Abs	963198	#EB27E	-	61	36					
%IF36	Abs	963683	#EB463	-	248	235					
%IF37	Abs	963738	#EB49A	-	266	252					
%IF38	Abs	963813	#EB4E5	-	294	286					
%IF39	Abs	963978	#EB58A	-	352	320					
%IF40	Abs	964513	#EB6D1	-	499	487					
%IF41	Abs	964483	#EB783	-	556	517					
%IF43	Abs	965064	#EB9C8	-	733	724					
%IF44	Abs	963438	#EB36E	-	161	152					
%RP1	Abs	963773	#EB4BD	-	282	209					
%RP2	Abs	964851	#EB8F3	-	676	604					
%TH313	Abs	963968	#EB580	-	346	343					
%TH36	Abs	963763	#EB4B3	-	276	244					
%TH37	Abs	963763	#EB4B3	-	272	262					
%TH39	Abs	963988	#EB594	-	361	348					
2DROP	Ext			-	302	564					
2DUP	Ext			-	478						
>R	Ext			-	141	523	656				
?DUP	Ext			-	34	318	721				
ADD	Ext			-	42	83	217	326	537	546	625	670
AND	Ext			-	150	338	635	640				
AT	Ext			-	82	192	200	202	205	214	216	257
					268	284	288	296	298	328	507	592
					599	627	637	648	792			
BRNCH	Ext			-	243	261	277	342	347	671		
C@+	Ext			-	156							
CAT	Ext			-	143	383	436					
CHR$	Ext			-	805							
CMOVE	Ext			-	529							
DEC	Ext			-	684							
DOCOL	Ext			-	23	73	92	124	137	170	313	372
					474	574						
DROP	Ext			-	113	126	157	464	494	525	565	677
					728							
DUP	Ext			-	43	223	327	387	435	440	512	522
					534	598	608	626				
EQUAL	Ext			-	145	390	397	405	419	443	450	601
FADJ	Ext			-	515							
FCREAT	Ext			-	33							
FILL	Ext			-	53							
FIVE+	Ext			-	335	632						
GT	Ext			-	203	639						
HEX	Ext			-	593							
LISTNG	Ext			-	747							
LIT	Ext			-	31	40	50	62	78	96	114	127
					174	230	324	376	388	395	403	417
					441	448	513	527	535	544	557	609
					623	668	688	803				

MAX	Ext			-	299						
MIN	Ext			-	98						
MINUS	Ext			-	649						
MULT	Ext			-	80						
N!	Ext			-	540						
N@	Ext			-	336	633					
NULL$	Ext			-	681	693	709	736	750	780	
ONE	Ext			-	54	188	337	345	500	538	634	638
ONE+	Ext			-	258						
ONE-	Ext			-	77	269					
OR	Ext			-	206	398	406	412	420	451	453	
OVER	Ext			-	142	149	382	394	402	410	628	
OVER2	Ext			-	249						
PSTOR	Ext			-	502	802					
QUOTC	Ext			-	27	105	578	700	714	729	741	756
					762	768	773	785				
R>	Ext			-	144	530	658					
ROT	Ext			-	424	425	452	622	655	662	663	
S!	Ext			-	100	585	683					
S<	Ext			-	250							
S<&	Ext			-	109	616	618	657	707	734	748	761
					767	772	778	794				
S=	Ext			-	233							
SEMI	Ext			-	57	65	84	116	129	162	290	304
					364	466	495	552	566	807		
STORE	Ext			-	45	56	187	190	194	225	242	260
					271	301	363	547				
SWAP	Ext			-	334	416	447	457	526	539		
SWAP2	Ext			-	615							
TWO+	Ext			-	458							
TWO/	Ext			-	218							
XDO	Ext			-	430	691						
XLOOP	Ext			-	459	695						
ZBRNH	Ext			-	35	151	208	234	251	285	319	339
					480	486	516	603	641	723		
ZEQ	Ext			-	207	465	602	722				
ZERO	Ext			-	185	303	353	354	426	690		
a&CLR	Ext			-	590							
=a>ADR	Abs	963218	#EB292	-	73	229	289	508				
a>PAD	Ext			-	705	719	746	790				
=a>SYT	Abs	964263	#EB6A7	-	474							
aBAS$	Ext			-	766	777						
aBFOUN	Ext			-	186	204	241	283				
aBLNK2	Ext			-	617							
aBLOW	Ext			-	189	199	215	259	297			
aBTOP	Ext			-	193	201	213	270				
aDIG10	Ext			-	411							
aDROP4	Ext			-	239							
=aDROPc	Abs	963378	#EB332	-	137	176	378					
aDUP4	Ext			-	232							
aEJECT	Ext			-	589							
=aENS6L	Abs	963348	#EB314	-	124	184						
=aENS8L	Abs	963263	#EB2BF	-	92	125						
aERCNT	Ext			-	791							
aERR$	Ext			-	99	101	524					

```
aFLTYP  Ext                 -   636
aKNOWN  Ext                 -   362
aLABOK  Abs  964003 #EB5A3  -   372   479
aLLIN#  Ext                 -   801
aLLINE  Ext                 -   614   752
aLUKUP  Abs  963443 #EB373  -   170   317   485
aMEMOV  Ext                 -   551
aNEG1   Ext                 -    52   240   341   600   800
aNUM#3  Ext                 -   793
aNUM#5  Ext                 -   654
aOBJFL  Ext                 -   647
aOFILE  Ext                 -   720
aOUTLI  Ext                 -   664   694   708   710   735   737   749   751
                                779   781   795   806
aOUTSY  Abs  964513 #EB7A1  -   574
aSFILE  Ext                 -   706
aSHUTD  Ext                 -    64   559
aSTIT   Ext                 -   584   682
aSYT>   Abs  963863 #EB517  -   313
aSYTAD  Ext                 -    44    81   591
aSYTIN  Abs  963072 #EB200  -    23
aSYTLS  Ext                 -   224   256   267   287   295   300   506
aSYTSZ  Ext                 -    55   191   501
zLEFT   Abs       0 #00000  -   810   811
zNEXT   Abs  965376 #EBB00  -     5   810
zSIZE   Abs    2304 #00900  -   809
zTHIS   Abs  963072 #EB200  -     4   809
```


Input Parameters

Source file name is GN&AS3

Listing file name is GN/AS3::65

Object file name is GN%AS3::65

```
                                111111
                       0123456789012345
```

Initial flag settings are

Errors

None

Saturn Assembler News


```
 1                      TITLE  AS4:_FORTH_ASSEMBLER_EXPRESSIONS
 2                      RDSYMB MR%GTO
 3 EBB00                ABS    #EBB00
 4              zTHIS   EQU    *
 5              zNEXT   EQU    #EC600
 6
 7              ************************************************
 8              ***
 9              *** MR&AS4     <840504.1037>
10              ***
11              ***
12              ***  FORTH ASSEMBLER
13              ***  LEXICAL ANALYSIS, EXPRESSION EVALUATION
14              ***
15              ************************************************
16              ****************************************************************
17              ***
18              ***  +DIGIT
19              ***
20              *** ( DIG BASE -- )
21              *** ( ADD A DIGIT TO THE VARIABLE VALUE )
22              ****************************************************************
23 EBB00 0000 =a+DIGI CON(5) =DOCOL          : +DIGIT
        0
24              *
25              * SHIFT VARIABLE VALUE ONE DIGIT
26              *
27 EBB05 0000         CON(5) =aVALUE         VALUE
        0
28 EBB0A 0000         CON(5) =AT             @
        0
29 EBB0F 0000         CON(5) =MULT           *
        0
30              *
31              * ADD IN NEW DIGIT
32              *
33 EBB14 0000         CON(5) =ADD            +           ( * NUM * )
        0
34 EBB19 0000         CON(5) =aVALUE         VALUE
        0
35 EBB1E 0000         CON(5) =STORE          !
        0
36 EBB23 0000         CON(5) =SEMI           ;           ( * * )
        0
37              ****************************************************************
38              ***
39              ***  ?DIGIT.10
40              ***
41              *** ( C -- F )
42              *** ( IS C A DECIMAL DIGIT ? )
43              ****************************************************************
44 EBB28 0000 =aDIG10 CON(5) =DOCOL          : ?DIGIT.10
        0
45              *
46              * IS C A DECIMAL DIGIT ?
```

```
47                *
48 EBB2D 0000             CON(5) =LIT            10
         0
49 EBB32 A000             CON(5) 10
         0
50 EBB37 0000             CON(5) =DIGIT          DIGIT
         0
51 EBB3C 0000             CON(5) =ZBRNH          IF          ( * [NUM] * )
         0
52 EBB41 4100             REL(5) %IF44
         0
53                *
54                * RETURN TRUE FLAG, NOT CONVERTED VALUE
55                *
56 EBB46 0000             CON(5) =DROP           DROP
         0
57 EBB4B 0000             CON(5) =aNEG1          -1          ( * TF * )
         0
58 EBB50 0000             CON(5) =SEMI           ;           ( * TF * )
         0
59                *
60                * RETURN FALSE FLAG.
61                *
62 EBB55                   %IF44
63 EBB55 0000             CON(5) =ZERO           0
         0
64 EBB5A 0000             CON(5) =SEMI           ;           ( * F * )
         0
65                ****************************************************************
66                ***
67                ***  CONSTANT.10
68                ***
69                *** ( STR1 -- STR2 1 )
70                *** ( CONVERT DECIMAL DIGITS IN STR1 INTO 'VALUE' )
71                ****************************************************************
72 EBB5F 0000 =aCON10 CON(5) =DOCOL          : CONSTANT.10
         0
73                *
74                * INITIALIZE VARIABLE VALUE
75                *
76 EBB64 0000             CON(5) =ZERO           0
         0
77 EBB69 0000             CON(5) =aVALUE         VALUE
         0
78 EBB6E 0000             CON(5) =STORE          !           ( * STR * )
         0
79                *
80                * while <CHARACTERS ARE DECIMAL DIGITS> do
81                *
82 EBB73                   %BG3                   BEGIN       ( * STR * )
83 EBB73 0000             CON(5) =OVER           OVER
         0
84 EBB78 0000             CON(5) =CAT            C@
         0
85 EBB7D 82BB             CON(5) =aDIG10         ?DIGIT.10   ( * STR F * )
```



```
         E
86 EBB82 0000             CON(5) =OVER           OVER
         0
87 EBB87 0000             CON(5) =AND            AND         ( * STR F * )
         0
88 EBB8C 0000             CON(5) =ZBRNH          WHILE       ( * STR * )
         0
89 EBB91 7300             REL(5) %RP3
         0
90                *
91                * EXTRACT DIGIT
92                *
93 EBB96 0000             CON(5) =C@+            C@+
         0
94 EBB9B 0000             CON(5) =LIT            10
         0
95 EBBA0 A000             CON(5) 10
         0
96 EBBA5 0000             CON(5) =DIGIT          DIGIT       ( * STR NUM TF * )
         0
97 EBBAA 0000             CON(5) =DROP           DROP
         0
98                *
99                * ADD VALUE TO VARIABLE VALUE
100               *
101 EBBAF 0000            CON(5) =LIT            10
         0
102 EBBB4 A000            CON(5) 10
         0
103 EBBB9 00BB            CON(5) =a+DIGI         +DIGIT      ( * STR * )
         E
104 EBBBE 0000            CON(5) =BRNCH          REPEAT
         0
105 EBBC3 0BFF            REL(5) %BG3
         F
106               *
107               * end while
108               *
109 EBBC8                  %RP3
110               *
111               * RETURN CONSTANT TOKEN (1) TO LEXSCAN
112               *
113 EBBC8 0000            CON(5) =ONE            1
         0
114 EBBCD 0000            CON(5) =SEMI           ;           ( * STR 1 * )
         0
115               ****************************************************************
116               ***
117               ***  ?DIGIT.16
118               ***
119               *** ( C -- F )
120               *** ( IS C A HEXIDECIMAL DIGIT ? )
121               ****************************************************************
122 EBBD2 0000 =aDIG16 CON(5) =DOCOL          : ?DIGIT.16
         0
```

```
123           *
124           * IS C A HEXIDECIMAL DIGIT ?
125           *
126 EBBD7 0000        CON(5) =LIT          16
          0
127 EBBDC 0100        CON(5) 16
          0
128 EBBE1 0000        CON(5) =DIGIT        DIGIT
          0
129 EBBE6 0000        CON(5) =ZBRNH        IF          ( * [NUM] * )
          0
130 EBBEB 4100        REL(5) %IF45
          0
131           *
132           * RETURN TRUE FLAG, NOT VALUE
133           *
134 EBBF0 0000        CON(5) =DROP         DROP
          0
135 EBBF5 0000        CON(5) =aNEG1        -1          ( * TF * )
          0
136 EBBFA 0000        CON(5) =SEMI         ;           ( * TF * )
          0
137           *
138           * RETURN FALSE FLAG.
139           *
140 EBBFF      %IF45
141 EBBFF 0000        CON(5) =ZERO         0
          0
142 EBC04 0000        CON(5) =SEMI         ;           ( * F * )
          0
143           ****************************************************************
144           ***
145           ***  CONSTANT.16
146           ***
147           *** ( STR1 -- STR2 1 )
148           *** ( CONVERT HEX DIGITS IN STR1 INTO 'VALUE' )
149           ****************************************************************
150 EBC09 0000 =aCON16 CON(5) =DOCOL        : CONSTANT.16
          0
151           *
152           * INITIALIZE VARIABLE VALUE
153           *
154 EBC0E 0000        CON(5) =ZERO         0
          0
155 EBC13 0000        CON(5) =aVALUE       VALUE
          0
156 EBC18 0000        CON(5) =STORE        !           ( * STR * )
          0
157           *
158           * while <CHARACTERS HEXIDECIMAL DIGITS> do
159           *
160 EBC1D      %BG4                        BEGIN       ( * STR * )
161 EBC1D 0000        CON(5) =OVER         OVER
          0
162 EBC22 0000        CON(5) =CAT          C@
```



```
          0
163 EBC27 2DBB        CON(5) =aDIG16       ?DIGIT.16   ( * STR F * )
          E
164 EBC2C 0000        CON(5) =OVER         OVER
          0
165 EBC31 0000        CON(5) =AND          AND         ( * STR F * )
          0
166 EBC36 0000        CON(5) =ZBRNH        WHILE       ( * STR * )
          0
167 EBC3B 7300        REL(5) %RP4
          0
168           *
169           * EXTRACT CHARACTER
170           *
171 EBC40 0000        CON(5) =C@+          C@+
          0
172 EBC45 0000        CON(5) =LIT          16
          0
173 EBC4A 0100        CON(5) 16
          0
174 EBC4F 0000        CON(5) =DIGIT        DIGIT       ( * STR NUM TF * )
          0
175 EBC54 0000        CON(5) =DROP         DROP
          0
176           *
177           * ADD DIGIT TO VARIABLE VALUE
178           *
179 EBC59 0000        CON(5) =LIT          16
          0
180 EBC5E 0100        CON(5) 16
          0
181 EBC63 00BB        CON(5) =a+DIGI       +DIGIT      ( * STR * )
          E
182 EBC68 0000        CON(5) =BRNCH        REPEAT
          0
183 EBC6D 0BFF        REL(5) %BG4
          F
184           *
185           * end while
186           *
187 EBC72      %RP4
188           *
189           * RETURN CONSTANT TOKEN (1) TO LEXSCAN
190           *
191 EBC72 0000        CON(5) =ONE          1
          0
192 EBC77 0000        CON(5) =SEMI         ;           ( * STR 1 * )
          0
193           ****************************************************************
194           ***
195           ***  CONSTANT.ASC
196           ***
197           *** ( STR1 -- STR2 1 )
198           *** ( CONVERT ASCII TO A VALUE )
199           ****************************************************************
```

```
200 EBC7C 0000 =aCONFF CON(5) =DOCOL         : CONSTANT.ASC
          0
201            *
202            * INITIALIZE VARIABLE VALUE
203            *
204 EBC81 0000         CON(5) =ZERO          0
          0
205 EBC86 0000         CON(5) =aVALUE        VALUE
          0
206 EBC8B 0000         CON(5) =STORE         !           ( * STR * )
          0
207            *
208            * while <CHARACTER # '> do
209            *
210 EBC90       %BG5                         BEGIN       ( * STR * )
211 EBC90 0000         CON(5) =OVER          OVER
          0
212 EBC95 0000         CON(5) =CAT           C@
          0
213 EBC9A 0000         CON(5) =LIT           39
          0
214 EBC9F 7200         CON(5) \'\
          0
215 EBCA4 0000         CON(5) =EQUAL         =           ( * STR F * )
          0
216 EBCA9 0000         CON(5) =ZEQ           0=
          0
217 EBCAE 0000         CON(5) =OVER          OVER
          0
218 EBCB3 0000         CON(5) =AND           AND         ( * STR F * )
          0
219 EBCB8 0000         CON(5) =ZBRNH         WHILE       ( * STR * )
          0
220 EBCBD 3200         REL(5) %RP5
          0
221            *
222            * EXTRACT CHARACTER
223            *
224 EBCC2 0000         CON(5) =C@+           C@+
          0
225            *
226            * ADD VALUE TO VARIABLE VALUE
227            *
228 EBCC7 0000         CON(5) =LIT           256
          0
229 EBCCC 0010         CON(5) 256
          0
230 EBCD1 00BB         CON(5) =a+DIGI        +DIGIT      ( * STR * )
          E
231 EBCD6 0000         CON(5) =BRNCH         REPEAT
          0
232 EBCDB 5BFF         REL(5) %BG5
          F
233            *
234            * end while
```



```
235            *
236 EBCE0       %RP5
237            *
238            * RETURN CONSTANT TOKEN (1) TO LEXSCAN
239            *
240 EBCE0 0000         CON(5) =ONE           1
          0
241 EBCE5 0000         CON(5) =SEMI          ;           ( * STR 1 * )
          0
242            ****************************************************************
243            ***
244            ***  ?SINGLE.TOKEN
245            ***
246            *** ( C -- F )
247            *** ( IS C ONE OF THE SINGLE TOKENS ? )
248            ****************************************************************
249 EBCEA 0000 =a?STOK CON(5) =DOCOL         : ?SINGLE.TOKEN
          0
250 EBCEF 0000         CON(5) =DUP           DUP         ( * C C * )
          0
251            *
252            * CHARACTER = BL ?  (SPACE)
253            *
254 EBCF4 0000         CON(5) =BL            BL
          0
255 EBCF9 0000         CON(5) =EQUAL         =           ( * C F * )
          0
256            *
257            * CHARACTER = ! ?   (BOOLEAN OR)
258            *
259 EBCFE 0000         CON(5) =OVER          OVER        ( * C F C * )
          0
260 EBD03 0000         CON(5) =LIT           33
          0
261 EBD08 1200         CON(5) \!\
          0
262 EBD0D 0000         CON(5) =EQUAL         =
          0
263 EBD12 0000         CON(5) =OR            OR          ( * C F * )
          0
264            *
265            * CHARACTER = & ?   (BOOLEAN AND)
266            *
267 EBD17 0000         CON(5) =OVER          OVER        ( * C F C * )
          0
268 EBD1C 0000         CON(5) =LIT           38
          0
269 EBD21 6200         CON(5) \&\
          0
270 EBD26 0000         CON(5) =EQUAL         =
          0
271 EBD2B 0000         CON(5) =OR            OR          ( * C F * )
          0
272            *
273            * CHARACTER = ( ?   (LEFT PAREN)
```

```
274                *
275 EBD30 0000     CON(5) =OVER          OVER        ( * C F C * )
          0
276 EBD35 0000     CON(5) =LIT           40
          0
277 EBD3A 8200     CON(5) \(\
          0
278 EBD3F 0000     CON(5) =EQUAL         =
          0
279 EBD44 0000     CON(5) =OR            OR          ( * C F * )
          0
280                *
281                * CHARACTER = ) ?   (RIGHT PAREN)
282                *
283 EBD49 0000     CON(5) =OVER          OVER        ( * C F C * )
          0
284 EBD4E 0000     CON(5) =LIT           41
          0
285 EBD53 9200     CON(5) \)\
          0
286 EBD58 0000     CON(5) =EQUAL         =
          0
287 EBD5D 0000     CON(5) =OR            OR          ( * C F * )
          0
288                *
289                * CHARACTER = * ?   (TIMES)
290                *
291 EBD62 0000     CON(5) =OVER          OVER        ( * C F C * )
          0
292 EBD67 0000     CON(5) =LIT           42
          0
293 EBD6C A200     CON(5) \*\
          0
294 EBD71 0000     CON(5) =EQUAL         =
          0
295 EBD76 0000     CON(5) =OR            OR          ( * C F * )
          0
296                *
297                * CHARACTER = + ?   (PLUS)
298                *
299 EBD7B 0000     CON(5) =OVER          OVER        ( * C F C * )
          0
300 EBD80 0000     CON(5) =LIT           43
          0
301 EBD85 B200     CON(5) \+\
          0
302 EBD8A 0000     CON(5) =EQUAL         =
          0
303 EBD8F 0000     CON(5) =OR            OR          ( * C F * )
          0
304                *
305                * CHARACTER = - ?   (MINUS)
306                *
307 EBD94 0000     CON(5) =OVER          OVER        ( * C F C * )
          0
```



```
308 EBD99 0000     CON(5) =LIT           45
          0
309 EBD9E D200     CON(5) \-\
          0
310 EBDA3 0000     CON(5) =EQUAL         =
          0
311 EBDA8 0000     CON(5) =OR            OR          ( * C F * )
          0
312                *
313                * CHARACTER = / ?   (DIVIDE)
314                *
315 EBDAD 0000     CON(5) =SWAP          SWAP        ( * F C * )
          0
316 EBDB2 0000     CON(5) =LIT           47
          0
317 EBDB7 F200     CON(5) \/\
          0
318 EBDBC 0000     CON(5) =EQUAL         =
          0
319 EBDC1 0000     CON(5) =OR            OR          ( * C F * )
          0
320                *
321                * IF ANY OF ABOVE QUESTIONS TRUE, THEN CHARACTER IS A SINGLE
322                * TOKEN.   RETURN THAT TOKEN TO LEXSCAN
323                *
324 EBDC6 0000     CON(5) =SEMI          ;           ( * F * )
          0
325                ************************************************************
326                ***
327                ***  ?END.LABEL
328                ***
329                *** ( C -- F )
330                *** ( IS C ONE OF THE CHARACTERS THAT MARKS END OF LABEL ? )
331                ************************************************************
332 EBDCB 0000 =a?ELAB CON(5) =DOCOL        : ?END.LABEL
          0
333                *
334                * CHARACTER A BL ?  (BLANK)
335                *
336 EBDD0 0000     CON(5) =DUP           DUP
          0
337 EBDD5 0000     CON(5) =BL            BL
          0
338 EBDDA 0000     CON(5) =EQUAL         =           ( * C F * )
          0
339                *
340                * CHARACTER A ) ?   (RIGHT PAREN)
341                *
342 EBDDF 0000     CON(5) =SWAP          SWAP        ( * F C * )
          0
343 EBDE4 0000     CON(5) =LIT           41
          0
344 EBDE9 9200     CON(5) \)\
          0
345 EBDEE 0000     CON(5) =EQUAL         =
```

```
          0
346 EBDF3 0000          CON(5) =OR              OR          ( * F * )
          0
347               *
348               * IF EITHER QUESTION IS TRUE, THEN RETURN TRUE.
349               *
350 EBDF8 0000          CON(5) =SEMI            ;           ( * F * )
          0
351               ****************************************************************
352               ***
353               ***  DO.LABEL
354               ***
355               *** ( STR1 -- STR2 2 )
356               *** ( PROCESS AN IDENTIFIER )
357               ****************************************************************
358 EBDFD 0000 =aDOLAB CON(5) =DOCOL           : DO.LABEL
          0
359               *
360               * SAVE BEGINNING ADDRESS
361               *
362 EBE02 0000          CON(5) =OVER            OVER
          0
363 EBE07 0000          CON(5) =>R              >R          ( * STR * )
          0
364               *
365               * while <NOT AT END OF LABEL> do
366               *
367 EBE0C          %BG6                         BEGIN       ( * STR * )
368 EBE0C 0000          CON(5) =OVER            OVER
          0
369 EBE11 0000          CON(5) =CAT             C@
          0
370 EBE16 BCDB          CON(5) =a?ELAB          ?END.LABEL  ( * STR F * )
          E
371 EBE1B 0000          CON(5) =ZEQ             0=
          0
372 EBE20 0000          CON(5) =OVER            OVER
          0
373 EBE25 0000          CON(5) =AND             AND         ( * STR F * )
          0
374 EBE2A 0000          CON(5) =ZBRNH           WHILE
          0
375 EBE2F 9100          REL(5) %RP6
          0
376               *
377               * DROP CHARACTERS FROM STRING
378               *
379 EBE34 0000          CON(5) =C@+             C@+
          0
380 EBE39 0000          CON(5) =DROP            DROP
          0
381 EBE3E 0000          CON(5) =BRNCH           REPEAT      ( * STR * )
          0
382 EBE43 9CFF          REL(5) %BG6
          F
```



```
383 EBE48          %RP6
384               *
385               * end while
386               *
387               *
388               * CALCULATE NUMBER OF CHARACTERS IN LABEL
389               *
390 EBE48 0000          CON(5) =OVER            OVER
          0
391 EBE4D 0000          CON(5) =R@              R@
          0
392 EBE52 0000          CON(5) =MINUS           -
          0
393 EBE57 0000          CON(5) =TWO/            2/
          0
394               *
395               * RECOVER ADDRESS FROM RETURN STACK
396               *
397 EBE5C 0000          CON(5) =R>              R>          ( * STR LEN ADDR * )
          0
398 EBE61 0000          CON(5) =SWAP            SWAP
          0
399               *
400               * SAVE STRING IN VARIABLE IDENTIFIER
401               *
402 EBE66 0000          CON(5) =LIT             10
          0
403 EBE6B A000          CON(5) 10
          0
404 EBE70 0000          CON(5) =MIN             MIN         ( * STR * )
          0
405 EBE75 0000          CON(5) =aIDENT          IDENTIFIER
          0
406 EBE7A 0000          CON(5) =S!              S!
          0
407               *
408               * RETURN IDENTIFIER TOKEN (2) TO LEXSCAN
409               *
410 EBE7F 0000          CON(5) =TWO             2
          0
411 EBE84 0000          CON(5) =SEMI            ;           ( * STR 2 * )
          0
412               ****************************************************************
413               ***
414               ***  LEXSCAN
415               ***
416               *** ( STR1 -- STR2 )
417               *** ( SCAN STRING FOR A TOKEN )
418               ****************************************************************
419 EBE89 0000 =aLEXSC CON(5) =DOCOL          : LEXSCAN
          0
420               *
421               * END OF STRING?
422               *
423 EBE8E 0000          CON(5) =DUP             DUP
```

```
          0
  424 EBE93 0000          CON(5) =ZEQ           0=
          0
  425 EBE98 0000          CON(5) =ZBRNH         IF
          0
  426 EBE9D 4100          REL(5) %IF46
          0
  427             *
  428             * YES.  PASS ON END TOKEN (0)
  429             *
  430 EBEA2 0000          CON(5) =ZERO          0           ( * STR 0 * )
          0
  431 EBEA7 0000          CON(5) =BRNCH         ELSE        ( * STR * )
          0
  432 EBEAC 5410          REL(5) %TH46
          0
  433             *
  434             * CHECK FOR DECIMAL CONSTANT
  435             *
  436 EBEB1        %IF46
  437 EBEB1 0000          CON(5) =OVER          OVER
          0
  438 EBEB6 0000          CON(5) =CAT           C@          ( * STR C * )
          0
  439 EBEBB 0000          CON(5) =DUP           DUP
          0
  440 EBEC0 82BB          CON(5) =aDIG10        ?DIGIT.10
          E
  441 EBEC5 0000          CON(5) =ZBRNH         IF
          0
  442 EBECA 9100          REL(5) %IF47
          0
  443 EBECF 0000          CON(5) =DROP          DROP
          0
  444 EBED4 F5BB          CON(5) =aCON10        CONSTANT.10
          E
  445 EBED9 0000          CON(5) =BRNCH         ELSE
          0
  446 EBEDE 3110          REL(5) %TH47
          0
  447             *
  448             * CHECK FOR HEXIDECIMAL CONSTANT (LEADING #)
  449             *
  450 EBEE3        %IF47
  451 EBEE3 0000          CON(5) =DUP           DUP
          0
  452 EBEE8 0000          CON(5) =LIT           35
          0
  453 EBEED 3200          CON(5) \#\
          0
  454 EBEF2 0000          CON(5) =EQUAL         =
          0
  455 EBEF7 0000          CON(5) =ZBRNH         IF
          0
  456 EBEFC 3200          REL(5) %IF48
```



```
          0
  457 EBF01 0000          CON(5) =DROP          DROP
          0
  458 EBF06 0000          CON(5) =C@+           C@+
          0
  459 EBF0B 0000          CON(5) =DROP          DROP
          0
  460 EBF10 90CB          CON(5) =aCON16        CONSTANT.16
          E
  461 EBF15 0000          CON(5) =BRNCH         ELSE
          0
  462 EBF1A 7D00          REL(5) %TH48
          0
  463             *
  464             * CHECK FOR ASCII CONSTANT (LEADING ')
  465             *
  466 EBF1F        %IF48
  467 EBF1F 0000          CON(5) =DUP           DUP
          0
  468 EBF24 0000          CON(5) =LIT           39
          0
  469 EBF29 7200          CON(5) \'\
          0
  470 EBF2E 0000          CON(5) =EQUAL         =
          0
  471 EBF33 0000          CON(5) =ZBRNH         IF
          0
  472 EBF38 F500          REL(5) %IF49
          0
  473 EBF3D 0000          CON(5) =DROP          DROP
          0
  474 EBF42 0000          CON(5) =C@+           C@+
          0
  475 EBF47 0000          CON(5) =DROP          DROP
          0
  476 EBF4C C7CB          CON(5) =aCONFF        CONSTANT.ASC
          E
  477 EBF51 0000          CON(5) =>R            >R          ( * STR * )
          0
  478 EBF56 0000          CON(5) =C@+           C@+
          0
  479 EBF5B 0000          CON(5) =LIT           39
          0
  480 EBF60 7200          CON(5) \'\
          0
  481 EBF65 0000          CON(5) =EQUAL         =
          0
  482 EBF6A 0000          CON(5) =ZEQ           0=
          0
  483 EBF6F 0000          CON(5) =ZBRNH         IF
          0
  484 EBF74 4100          REL(5) %IF43
          0
  485 EBF79 0000          CON(5) =LIT           24
          0
```

```
486 EBF7E 8100          CON(5) 24               'invalid quoted string'
          0
487 EBF83 0000          CON(5) =aERROR          ERROR
          0
488 EBF88       %IF43                          THEN
489 EBF88 0000          CON(5) =R>              R>          ( * STR TOK * )
          0
490 EBF8D 0000          CON(5) =BRNCH           ELSE
          0
491 EBF92 F500          REL(5) %TH49
          0
492             *
493             * CHECK FOR SINGLE TOKEN (ALL OPERATORS & PARENTHESIS)
494             *
495 EBF97        %IF49
496 EBF97 0000          CON(5) =DUP             DUP
          0
497 EBF9C AECB          CON(5) =a?STOK          ?SINGLE.TOKEN
          E
498 EBFA1 0000          CON(5) =ZBRNH           IF
          0
499 EBFA6 9100          REL(5) %IF50
          0
500 EBFAB 0000          CON(5) =DROP            DROP
          0
501 EBFB0 0000          CON(5) =C@+             C@+
          0
502 EBFB5 0000          CON(5) =BRNCH           ELSE
          0
503 EBFBA 7300          REL(5) %TH50
          0
504             *
505             * CHECK FOR LABEL
506             *
507 EBFBF        %IF50
508 EBFBF 0000          CON(5) =DUP             DUP
          0
509 EBFC4 0000          CON(5) =LIT             61
          0
510 EBFC9 D300          CON(5) \=\
          0
511 EBFCE 0000          CON(5) =EQUAL           =
          0
512 EBFD3 0000          CON(5) =ZBRNH           IF
          0
513 EBFD8 F000          REL(5) %IF51
          0
514 EBFDD 0000          CON(5) =DROP            DROP
          0
515 EBFE2 0000          CON(5) =C@+             C@+
          0
516             *
517             * NONE OF ABOVE.  ASSUME LABEL, AND LET IT GENERATE ANY ERRORS.
518             *
519 EBFE7        %IF51
```



```
520 EBFE7 0000          CON(5) =DROP            DROP
          0
521 EBFEC DFDB          CON(5) =aDOLAB          DO.LABEL
          E
522             *
523             * STORE TOKEN IN VARIABLE TOKEN
524             *
525 EBFF1        %TH50                          THEN        ( * STR NUM * )
526 EBFF1        %TH49                          THEN
527 EBFF1        %TH48                          THEN
528 EBFF1        %TH47                          THEN        ( * STR NUM * )
529 EBFF1        %TH46                          THEN
530 EBFF1 0000          CON(5) =aTOKEN          TOKEN
          0
531 EBFF6 0000          CON(5) =STORE           !
          0
532 EBFFB 0000          CON(5) =SEMI            ;           ( * STR * )
          0
533             ****************************************************************
534             ***
535             ***  BASE MODULE
536             ***
537             *** ( STR -- STR NUM )
538             *** ( RECURSIVE DESCENT: MODULE identifiers )
539             ****************************************************************
540 EC000 0000 =amBASE CON(5) =DOCOL        : BASE MODULE
          0
541             *
542             * ENSURE WE'RE NOT OVERFLOWING ANY STACKS
543             *
544 EC005 CF3C          CON(5) =aDEPCK          DEPTH CHECK
          E
545             *
546             * select TOKEN
547             *
548 EC00A 0000          CON(5) =aTOKEN          TOKEN
          0
549 EC00F 0000          CON(5) =AT              @ CASE
          0
550             *
551             * case 1 of
552             *
553 EC014 0000          CON(5) =ONE             1
          0
554 EC019 0000          CON(5) =OVER            OF
          0
555 EC01E 0000          CON(5) =EQUAL
          0
556 EC023 0000          CON(5) =ZBRNH
          0
557 EC028 E100          REL(5) %OF1
          0
558             *
559             * TOKEN IS A CONSTANT, RETURN IT'S VALUE
560             *
```

```
561 EC02D 0000          CON(5) =DROP                    ( * STR * )
          0
562 EC032 0000          CON(5) =aVALUE          VALUE
          0
563 EC037 0000          CON(5) =AT              @       ( * STR NUM * )
          0
564 EC03C 0000          CON(5) =BRNCH           ENDOF
          0
565 EC041 D110          REL(5) %EC1
          0
566             *
567             * case 2 of
568             *
569 EC046       %OF1
570 EC046 0000          CON(5) =TWO             2
          0
571 EC04B 0000          CON(5) =OVER            OF
          0
572 EC050 0000          CON(5) =EQUAL
          0
573 EC055 0000          CON(5) =ZBRNH
          0
574 EC05A 6400          REL(5) %OF2
          0
575             *
576             * TOKEN IS AN IDENTIFIER, RETURN IT'S VALUE
577             *
578 EC05F 0000          CON(5) =DROP                    ( * STR * )
          0
579 EC064 0000          CON(5) =aIDENT          IDENTIFIER
          0
580 EC069 0000          CON(5) =aSYT>           SYT>    ( * STR NUM * )
          0
581 EC06E 0000          CON(5) =aKNOWN          KNOWN
          0
582 EC073 0000          CON(5) =AT              @
          0
583 EC078 0000          CON(5) =ZEQ             0=
          0
584 EC07D 0000          CON(5) =ZBRNH           IF      ( * STR NUM * )
          0
585 EC082 4100          REL(5) %IF52
          0
586             *
587             * VALUE OF IDENTIFIER NOT KNOWN
588             *
589 EC087 0000          CON(5) =LIT             37
          0
590 EC08C 5200          CON(5) 37               'unrecognized label'
          0
591 EC091 A05C          CON(5) =aP2ERR          PASS 2 ERROR
          E
592 EC096       %IF52                           THEN    ( * STR NUM * )
593 EC096 0000          CON(5) =BRNCH           ENDOF
          0
```



```
594 EC09B 3C00          REL(5) %EC1
          0
595             *
596             * case 40 of
597             *
598 EC0A0       %OF2
599 EC0A0 0000          CON(5) =LIT             40
          0
600 EC0A5 8200          CON(5) \(\
          0
601 EC0AA 0000          CON(5) =OVER            OF
          0
602 EC0AF 0000          CON(5) =EQUAL
          0
603 EC0B4 0000          CON(5) =ZBRNH
          0
604 EC0B9 5500          REL(5) %OF3
          0
605             *
606             * TOKEN IS LEFT PARENETHESIS, MAKE RECURSIVE CALL BACK TO TOP LEV
607             *
608 EC0BE 0000          CON(5) =DROP                    ( * STR * )
          0
609 EC0C3 98EB          CON(5) =aLEXSC          LEXSCAN
          E
610 EC0C8 843C          CON(5) =amTERM          TERM    ( * STR NUM * )
          E
611             *
612             * ENSURE TOKEN AFTER RECURSIVE CALL IS A RIGHT PARENTHESIS
613             *
614 EC0CD 0000          CON(5) =aTOKEN          TOKEN
          0
615 EC0D2 0000          CON(5) =AT              @       ( * STR NUM TOK * )
          0
616 EC0D7 0000          CON(5) =LIT             41
          0
617 EC0DC 9200          CON(5) \)\
          0
618 EC0E1 0000          CON(5) =EQUAL           =
          0
619 EC0E6 0000          CON(5) =ZEQ             0=
          0
620 EC0EB 0000          CON(5) =ZBRNH           IF      ( * STR NUM * )
          0
621 EC0F0 4100          REL(5) %IF53
          0
622             *
623             * REPORT MISMATCHED PARENTHESES
624             *
625 EC0F5 0000          CON(5) =LIT             38
          0
626 EC0FA 6200          CON(5) 38               'mismatched parentheses'
          0
627 EC0FF A05C          CON(5) =aP2ERR          PASS 2 ERROR ( * STR NUM * )
          E
```

```
628 EC104       %IF53                       THEN
629 EC104 0000          CON(5) =BRNCH       ENDOF
          0
630 EC109 5500          REL(5) %EC1
          0
631             *
632             * case 42 of
633             *
634 EC10E       %OF3
635 EC10E 0000          CON(5) =LIT         42
          0
636 EC113 A200          CON(5) \*\
          0
637 EC118 0000          CON(5) =OVER        OF
          0
638 EC11D 0000          CON(5) =EQUAL
          0
639 EC122 0000          CON(5) =ZBRNH
          0
640 EC127 E100          REL(5) %OF4
          0
641             *
642             * TOKEN IS *, RETURN CURRENT LOCATION COUNTER
643             *
644 EC12C 0000          CON(5) =DROP                    ( * STR * )
          0
645 EC131 0000          CON(5) =aLC         LC
          0
646 EC136 0000          CON(5) =AT          @           ( * STR NUM * )
          0
647 EC13B 0000          CON(5) =BRNCH       ENDOF
          0
648 EC140 E100          REL(5) %EC1
          0
649             *
650             * case ELSE
651             *
652 EC145       %OF4
653             *
654             * TOKEN IS SOMETHING ELSE,  REPORT EXPRESSION SYNTAX ERROR
655             *
656 EC145 0000          CON(5) =DROP        DROP
          0
657 EC14A 0000          CON(5) =LIT         39
          0
658 EC14F 7200          CON(5) 39           'illegal expression'
          0
659 EC154 A05C          CON(5) =aP2ERR      PASS 2 ERROR
          E
660 EC159 0000          CON(5) =ZERO        0
          0
661             *
662             * end case
663             *
664 EC15E       %EC1                        ENDCASE     ( * STR NUM * )
```



```
665             *
666             * DO A PRE-SCAN FOR NEXT ROUTINE
667             *
668 EC15E 0000          CON(5) =>R          >R
          0
669 EC163 98EB          CON(5) =aLEXSC      LEXSCAN
          E
670 EC168 0000          CON(5) =R>          R>
          0
671 EC16D 0000          CON(5) =SEMI        ;           ( * STR NUM * )
          0
672             ****************************************************************
673             ***
674             ***  UNARY MINUS MODULE
675             ***
676             *** ( STR -- STR NUM )
677             *** ( RECURSIVE DESCENT: MODULE unary - )
678             ****************************************************************
679 EC172 0000 =amUN-  CON(5) =DOCOL       : UNARY MINUS MODULE
          0
680             *
681             * ENSURE WE'RE NOT OVERFLOWING ANY STACKS
682             *
683 EC177 CF3C          CON(5) =aDEPCK      DEPTH CHECK
          E
684             *
685             * IS CURRENT TOKEN A MINUS SIGN (-) ?
686             *
687 EC17C 0000          CON(5) =aTOKEN      TOKEN
          0
688 EC181 0000          CON(5) =AT          @
          0
689 EC186 0000          CON(5) =LIT         45
          0
690 EC18B D200          CON(5) \-\
          0
691 EC190 0000          CON(5) =EQUAL       =
          0
692 EC195 0000          CON(5) =ZBRNH       IF          ( * STR * )
          0
693 EC19A 9100          REL(5) %IF55
          0
694             *
695             * YES.  CONTINUE THROUGH DESCENT, THEN NEGATE RESULT
696             *
697 EC19F 98EB          CON(5) =aLEXSC      LEXSCAN     ( * STR * )
          E
698 EC1A4 000C          CON(5) =amBASE      BASE        ( * STR NUM * )
          E
699 EC1A9 0000          CON(5) =NEGAT       NEGATE      ( * STR NUM * )
          0
700 EC1AE 0000          CON(5) =SEMI        ;           ( * STR * )
          0
701             *
702             * NO.  CONTINUE THROUGH DESCENT.
```

```
                *
                %IF55                           ELSE        ( * STR * )
EC1B3           CON(5) =amBASE                  BASE        ( * STR NUM * )
EC1B3 000CE
EC1B8 00000     CON(5) =SEMI                    ;           ( * STR NUM * )
                ****************************************************************
                ***
                ***  BOOLEAN MODULE
                ***
                *** ( STR -- STR NUM )
                *** ( RECURSIVE DESCENT: MODULE &! )
                ****************************************************************
EC1BD 00000 =amBOOL CON(5) =DOCOL               : BOOLEAN MODULE
                *
                * ENSURE WE'RE NOT OVERFLOWING ANY STACKS
                *
EC1C2 CF3CE     CON(5) =aDEPCK                  DEPTH CHECK
                *
                * DO THE DESCENT
                *
EC1C7 271CE     CON(5) =amUN-                   UNARY -     ( * STR NUM * )
                *
                * while <TOKEN = &> or <TOKEN = !> do
                *
EC1CC           %BG7                            BEGIN       ( * STR NUM * )
EC1CC 00000     CON(5) =aTOKEN                  TOKEN
EC1D1 00000     CON(5) =AT                      @           ( * STR NUM TOK * )
EC1D6 00000     CON(5) =DUP                     DUP
EC1DB 00000     CON(5) =LIT                     38
EC1E0 62000     CON(5) \&\
EC1E5 00000     CON(5) =EQUAL                   =           ( * STR NUM TOK F * )
EC1EA 00000     CON(5) =OVER                    OVER
EC1EF 00000     CON(5) =LIT                     33
EC1F4 12000     CON(5) \!\
EC1F9 00000     CON(5) =EQUAL                   =
EC1FE 00000     CON(5) =OR                      OR          ( * STR NUM TOK F * )
EC203 00000     CON(5) =ZBRNH                   WHILE       ( * STR NUM TOK * )
EC208 F5000     REL(5) %RP7
```



```
                * "
                * SAVE CURRENT VALUE AND CURRENT TOKEN ON DATA STACK
                *
EC20D 00000     CON(5) =SWAP2                   2SWAP
                *
                * CONTINUE DESCENT
                *
EC212 98EBE     CON(5) =aLEXSC                  LEXSCAN     ( * NUM TOK STR * )
EC217 271CE     CON(5) =amUN-                   UNARY -
                *
                * PERFORM OPERATION ON NEWLY ACQUIRED NUMBER
                *
EC21C 00000     CON(5) =>R                      >R          ( * NUM TOK STR * )
EC221 00000     CON(5) =SWAP2                   2SWAP       ( * STR NUM TOK * )
EC226 00000     CON(5) =LIT                     38
EC22B 62000     CON(5) \&\
EC230 00000     CON(5) =EQUAL                   =           ( * STR NUM * )
EC235 00000     CON(5) =ZBRNH                   IF
EC23A 91000     REL(5) %IF54
EC23F 00000     CON(5) =R>                      R>          ( * STR NUM NUM' * )
EC244 00000     CON(5) =AND                     AND
EC249 00000     CON(5) =BRNCH
EC24E E7FFF     REL(5) %BG7                     REPEAT
EC253           %IF54                           ELSE        ( * STR NUM * )
EC253 00000     CON(5) =R>                      R>          ( * STR NUM NUM' * )
EC258 00000     CON(5) =OR                      OR
EC25D 00000     CON(5) =BRNCH                   REPEAT
EC262 A6FFF     REL(5) %BG7
                *
                * end while
                *
EC267           %RP7
EC267 00000     CON(5) =DROP                    DROP
EC26C 00000     CON(5) =SEMI                    ;           ( * STR NUM * )
                ****************************************************************
```

```
                ***
                ***  FACTOR MODULE
                ***
                *** ( STR -- STR NUM )
                *** ( RECURSIVE DESCENT: MODULE */ )
                ****************************************************************
EC271 0000 =amFACT CON(5) =DOCOL          : FACTOR MODULE
           0
                *
                * ENSURE WE'RE NOT OVERFLOWING ANY STACKS
                *
EC276 CF3C          CON(5) =aDEPCK        DEPTH CHECK
           E
                *
                * DO DESCENT
                *
EC27B DB1C          CON(5) =amBOOL        BOOLEAN     ( * STR NUM * )
           E
                *
                * while <TOKEN = *> or <TOKEN = /> do
                *
EC280          %BG9                       BEGIN       ( * STR NUM * )
EC280 0000          CON(5) =aTOKEN        TOKEN
           0
EC285 0000          CON(5) =AT            @           ( * STR NUM TOK * )
           0
EC28A 0000          CON(5) =DUP           DUP
           0
EC28F 0000          CON(5) =LIT           42
           0
EC294 A200          CON(5) \*\
           0
EC299 0000          CON(5) =EQUAL         =           ( * STR NUM TOK F * )
           0
EC29E 0000          CON(5) =OVER          OVER
           0
EC2A3 0000          CON(5) =LIT           47
           0
EC2A8 F200          CON(5) \/\
           0
EC2AD 0000          CON(5) =EQUAL         =
           0
EC2B2 0000          CON(5) =OR            OR          ( * STR NUM TOK F * )
           0
EC2B7 0000          CON(5) =ZBRNH         WHILE       ( * STR NUM TOK * )
           0
EC2BC 2800          REL(5) %RP9
           0
                *
                * SAVE CURRENT VALUE AND CURRENT TOKEN ON DATA STACK
                *
EC2C1 0000          CON(5) =SWAP2         2SWAP
           0
                *
                * CONTINUE DESCENT
```



```
                *
EC2C6 98EB          CON(5) =aLEXSC        LEXSCAN     ( * NUM TOK STR * )
           E
EC2CB DB1C          CON(5) =amBOOL        BOOLEAN
           E
                *
                * DO OPERATION ON NEWLY ACQUIRED VALUE
                *
EC2D0 0000          CON(5) =>R            >R
           0
EC2D5 0000          CON(5) =SWAP2         2SWAP       ( * STR NUM TOK * )
           0
EC2DA 0000          CON(5) =LIT           42
           0
EC2DF A200          CON(5) \*\
           0
EC2E4 0000          CON(5) =EQUAL         =
           0
EC2E9 0000          CON(5) =ZBRNH         IF
           0
EC2EE 9100          REL(5) %IF56
           0
EC2F3 0000          CON(5) =R>            R>          ( * STR NUM NUM' * )
           0
EC2F8 0000          CON(5) =MULT          *
           0
EC2FD 0000          CON(5) =BRNCH         REPEAT
           0
EC302 E7FF          REL(5) %BG9
           F
EC307          %IF56                      ELSE        ( * STR NUM * )
EC307 0000          CON(5) =R>            R>          ( * STR NUM NUM' * )
           0
                *
                * CHECK FOR DIVIDE BY ZERO
                *
EC30C 0000          CON(5) =DUP           DUP
           0
EC311 0000          CON(5) =ZEQ           0=
           0
EC316 0000          CON(5) =ZBRNH         IF          ( * STR NUM NUM' * )
           0
EC31B 4100          REL(5) %IF61
           0
EC320 0000          CON(5) =LIT           39
           0
EC325 7200          CON(5) 39             'illegal expression'
           0
EC32A A05C          CON(5) =aP2ERR        ERROR
           E
EC32F          %IF61                      THEN        ( * STR NUM NUM' * )
EC32F 0000          CON(5) =/             /
           0
EC334 0000          CON(5) =BRNCH         REPEAT
           0
```

```
845 EC339 74FF          REL(5) %BG9
          F
846             *
847             * end while
848             *
849 EC33E        %RP9
850 EC33E 0000          CON(5) =DROP            DROP
          0
851 EC343 0000          CON(5) =SEMI            ;           ( * STR NUM * )
          0
852             ****************************************************************
853             ***
854             ***  TERM MODULE
855             ***
856             *** ( STR -- STR NUM )
857             *** ( RECURSIVE DESCENT: MODULE +- )
858             ****************************************************************
859 EC348 0000 =amTERM CON(5) =DOCOL           : TERM MODULE
          0
860             *
861             * ENSURE WE'RE NOT OVERFLOWING ANY STACKS
862             *
863 EC34D CF3C          CON(5) =aDEPCK          DEPTH CHECK
          E
864             *
865             * DO DESCENT
866             *
867 EC352 172C          CON(5) =amFACT          FACTOR      ( * STR NUM * )
          E
868             *
869             * while <TOKEN = +> or <TOKEN = -> do
870             *
871 EC357        %BG11                          BEGIN       ( * STR NUM * )
872 EC357 0000          CON(5) =aTOKEN          TOKEN
          0
873 EC35C 0000          CON(5) =AT              @           ( * STR NUM TOK * )
          0
874 EC361 0000          CON(5) =DUP             DUP
          0
875 EC366 0000          CON(5) =LIT             43
          0
876 EC36B B200          CON(5) \+\
          0
877 EC370 0000          CON(5) =EQUAL           =           ( * STR NUM TOK F * )
          0
878 EC375 0000          CON(5) =OVER            OVER
          0
879 EC37A 0000          CON(5) =LIT             45
          0
880 EC37F D200          CON(5) \-\
          0
881 EC384 0000          CON(5) =EQUAL           =
          0
882 EC389 0000          CON(5) =OR              OR          ( * STR NUM TOK F * )
          0
```



```
883 EC38E 0000          CON(5) =ZBRNH           WHILE       ( * STR NUM TOK * )
          0
884 EC393 F500          REL(5) %RP11
          0
885             *
886             * SAVE CURRENT VALUE AND CURRENT TOKEN ON DATA STACK
887             *
888 EC398 0000          CON(5) =SWAP2           2SWAP
          0
889             *
890             * CONTINUE DESCENT
891             *
892 EC39D 98EB          CON(5) =aLEXSC          LEXSCAN     ( * NUM TOK STR * )
          E
893 EC3A2 172C          CON(5) =amFACT          FACTOR
          E
894 EC3A7 0000          CON(5) =>R              >R          ( * NUM TOK STR * )
          0
895             *
896             * PERFORM OPPERATION ON NEWLY ACQUIRED NUMBER
897             *
898 EC3AC 0000          CON(5) =SWAP2           2SWAP       ( * STR NUM TOK * )
          0
899 EC3B1 0000          CON(5) =LIT             43
          0
900 EC3B6 B200          CON(5) \+\
          0
901 EC3BB 0000          CON(5) =EQUAL           =           ( * STR NUM * )
          0
902 EC3C0 0000          CON(5) =ZBRNH           IF
          0
903 EC3C5 9100          REL(5) %IF57
          0
904 EC3CA 0000          CON(5) =R>              R>          ( * STR NUM NUM' * )
          0
905 EC3CF 0000          CON(5) =ADD             +
          0
906 EC3D4 0000          CON(5) =BRNCH           REPEAT
          0
907 EC3D9 E7FF          REL(5) %BG11
          F
908 EC3DE        %IF57                          ELSE        ( * STR NUM * )
909 EC3DE 0000          CON(5) =R>              R>          ( * STR NUM NUM' * )
          0
910 EC3E3 0000          CON(5) =MINUS           -
          0
911 EC3E8 0000          CON(5) =BRNCH           REPEAT
          0
912 EC3ED A6FF          REL(5) %BG11
          F
913             *
914             * end while
915             *
916 EC3F2        %RP11
917 EC3F2 0000          CON(5) =DROP            DROP
```

```
          0
  918 EC3F7 0000        CON(5) =SEMI          ;           ( * STR NUM * )
          0
  919           ************************************************************
  920           ***
  921           ***  DEPTH CHECK
  922           ***
  923           *** ( -- )
  924           *** ( CHECK THE RETURN STACK TO SEE IF IT IS OVERFLOWING INTO )
  925           *** ( THE TERMINAL INPUT BUFFER.  IF IT IS, CLEAN UP THE RETURN )
  926           *** ( AND DATA STACKS TO REFLECT AN ABNORMAL END TO 'EXPRESSION'
  927           *** ( AND REPORT 'illegal expression' )
  928           ************************************************************
  929 EC3FC 0000 =aDEPCK CON(5) =DOCOL         : DEPTH CHECK
          0
  930 EC401 0000        CON(5) =RP@           RP@         ( * PTR * )
          0
  931 EC406 0000        CON(5) =TIB           TIB
          0
  932 EC40B 0000        CON(5) =LIT           234
          0
  933 EC410 AE00        CON(5) (96+1)*2+7*5+5
          0
  934           *
  935           * NINETY SIX (96) CHARS IN TIB + ONE (1) MORE FOR NULL
  936           * EACH MODULE IN THE DESCENT CALLS SEVEN (7) LEVELS OF
  937           * RETURN STACK WHEN IT CALLS LEXSCAN.  EACH LEVEL REQUIRES
  938           * FIVE (5) NIBBLES.  AN EXTRA FIVE (5) FOR ENSURING NO OVERFLOW.
  939           *
  940 EC415 0000        CON(5) =ADD           +
          0
  941 EC41A 0000        CON(5) =LT            <
          0
  942 EC41F 0000        CON(5) =ZBRNH         IF
          0
  943 EC424 D700        REL(5) %IF60
          0
  944           *
  945           * RESET THE RETURN STACK
  946           *
  947 EC429      %BG1                         BEGIN       ( * * )
  948 EC429 0000        CON(5) =RP@           RP@
          0
  949 EC42E 0000        CON(5) =aRSTK         RTNSTK
          0
  950 EC433 0000        CON(5) =AT            @
          0
  951 EC438 0000        CON(5) =LT            <
          0
  952 EC43D 0000        CON(5) =ZBRNH         WHILE
          0
  953 EC442 9100        REL(5) %RP1
          0
  954 EC447 0000        CON(5) =R>            R>
          0
```



```
  955 EC44C 0000        CON(5) =DROP          DROP
          0
  956 EC451 0000        CON(5) =BRNCH         REPEAT
          0
  957 EC456 3DFF        REL(5) %BG1
          F
  958 EC45B      %RP1
  959           *
  960           * RESET THE DATA STACK
  961           *
  962 EC45B      %BG2                         BEGIN       ( * * )
  963 EC45B 0000        CON(5) =SP@           SP@
          0
  964 EC460 0000        CON(5) =aDSTK         DATASTK
          0
  965 EC465 0000        CON(5) =AT            @
          0
  966 EC46A 0000        CON(5) =LT            <
          0
  967 EC46F 0000        CON(5) =ZBRNH         WHILE
          0
  968 EC474 4100        REL(5) %RP2
          0
  969 EC479 0000        CON(5) =DROP          DROP
          0
  970 EC47E 0000        CON(5) =BRNCH         REPEAT
          0
  971 EC483 8DFF        REL(5) %BG2
          F
  972 EC488      %RP2                                   ( * STR * )
  973           *
  974           * AT THIS POINT, BOTH RETURN AND DATA STACKS SHOULD BE RETURNED
  975           * TO THEIR ORIGINAL (CALLING ROUTINE'S POSITION).  WE WANT TO
  976           * REMOVE THE ORIGINAL STRING FROM THE STACK AND RETURN A ZERO
  977           *
  978 EC488 0000        CON(5) =2DROP         2DROP       ( * * )
          0
  979 EC48D 0000        CON(5) =ZERO          0           ( * NUM * )
          0
  980 EC492 0000        CON(5) =LIT           39
          0
  981 EC497 7200        CON(5) 39             'illegal expression'
          0
  982 EC49C A05C        CON(5) =aP2ERR        PASS 2 ERROR
          E
  983 EC4A1      %IF60                        ELSE        ( * * )
  984 EC4A1 0000        CON(5) =SEMI          ;           ( * * )
          0
  985           ************************************************************
  986           ***
  987           ***  EXPRESSION
  988           ***
  989           *** ( STR -- NUM )
  990           *** ( EVALUATE THE EXPRESSION IN THE STRING )
  991           ************************************************************
```

```
 992 EC4A6 0000 =aEXPRS CON(5) =DOCOL          : EXPRESSION
          0
 993            *
 994            * SAVE ORIGINAL RETURN STACK
 995            *
 996 EC4AB 0000         CON(5) =RP@            RP@
          0
 997 EC4B0 0000         CON(5) =aRSTK          RTNSTK
          0
 998 EC4B5 0000         CON(5) =STORE          !
          0
 999            *
1000            * SAVE ORIGINAL DATA POINTER
1001            *
1002 EC4BA 0000         CON(5) =SP@            SP@
          0
1003 EC4BF 0000         CON(5) =aDSTK          DATASTK
          0
1004 EC4C4 0000         CON(5) =STORE          !
          0
1005            *
1006            * DO RECURSIVE DESCENT
1007            *
1008 EC4C9 98EB         CON(5) =aLEXSC         LEXSCAN
          E
1009 EC4CE 843C         CON(5) =amTERM         TERM          ( * STR NUM * )
          E
1010            *
1011            * REMOVE ORIGINAL STRING FROM STACK
1012            *
1013 EC4D3 0000         CON(5) =>R             >R
          0
1014 EC4D8 0000         CON(5) =2DROP          2DROP
          0
1015 EC4DD 0000         CON(5) =R>             R>            ( * NUM * )
          0
1016            *
1017            * CHECK FOR EXCESS CHARACTERS
1018            *
1019 EC4E2 0000         CON(5) =aTOKEN         TOKEN
          0
1020 EC4E7 0000         CON(5) =AT             @
          0
1021 EC4EC 0000         CON(5) =ZBRNH          IF            ( * NUM * )
          0
1022 EC4F1 4100         REL(5) %IF58
          0
1023 EC4F6 0000         CON(5) =LIT            40
          0
1024 EC4FB 8200         CON(5) 40              'excess characters in expression'
          0
1025 EC500 A05C         CON(5) =aP2ERR         PASS 2 ERROR
          E
1026 EC505       %IF58                         THEN
1027 EC505 0000         CON(5) =SEMI           ;             ( * NUM * )
```



```
          0
1028            ***************************************************************
1029            ***
1030            ***  PASS 2 ERROR
1031            ***
1032            *** ( NUM --- )
1033            *** ( REPORT THE ERROR IF IN PASS 2 ONLY )
1034            ***************************************************************
1035 EC50A 0000 =aP2ERR CON(5) =DOCOL          : PASS 2 ERROR
          0
1036            *
1037            * CHECK IF WE'RE PASS 1
1038            *
1039 EC50F 0000         CON(5) =a?PAS1         ?PASS=1
          0
1040 EC514 0000         CON(5) =ZBRNH          IF            ( * NUM * )
          0
1041 EC519 F000         REL(5) %IF59
          0
1042            *
1043            * YES, DO NOTHING FOR NOW.
1044            *
1045 EC51E 0000         CON(5) =DROP           DROP
          0
1046 EC523 0000         CON(5) =SEMI           ;             ( * * )
          0
1047            *
1048            * NO, ERRORS SHOULD BE REPORTED NOW.
1049            *
1050 EC528       %IF59                         ELSE          ( * NUM * )
1051 EC528 0000         CON(5) =aERROR         ERROR
          0
1052 EC52D 0000         CON(5) =SEMI           ;             ( * * )
          0
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER

```
1053                    STITLE FIT STATS
1054            zSIZE   EQU    (*)-zTHIS
1055            zLEFT   EQU    (zNEXT)-*
1056 EC532              BSS    zLEFT
1057 EC600              END
```



```
%BG1     Abs  967721 #EC429 -   947   957
%BG11    Abs  967511 #EC357 -   871   907   912
%BG2     Abs  967771 #EC45B -   962   971
%BG3     Abs  965491 #EBB73 -    82   105
%BG4     Abs  965661 #EBC1D -   160   183
%BG5     Abs  965776 #EBC90 -   210   232
%BG6     Abs  966156 #EBE0C -   367   382
%BG7     Abs  967116 #EC1CC -   726   762   767
%BG9     Abs  967296 #EC280 -   793   829   845
%EC1     Abs  967006 #EC15E -   664   565   594   630   648
%IF43    Abs  966536 #EBF88 -   488   484
%IF44    Abs  965461 #EBB55 -    62    52
%IF45    Abs  965631 #EBBFF -   140   130
%IF46    Abs  966321 #EBEB1 -   436   426
%IF47    Abs  966371 #EBEE3 -   450   442
%IF48    Abs  966431 #EBF1F -   466   456
%IF49    Abs  966551 #EBF97 -   495   472
%IF50    Abs  966591 #EBFBF -   507   499
%IF51    Abs  966631 #EBFE7 -   519   513
%IF52    Abs  966806 #EC096 -   592   585
%IF53    Abs  966916 #EC104 -   628   621
%IF54    Abs  967251 #EC253 -   763   758
%IF55    Abs  967091 #EC1B3 -   704   693
%IF56    Abs  967431 #EC307 -   830   825
%IF57    Abs  967646 #EC3DE -   908   903
%IF58    Abs  967941 #EC505 -  1026  1022
%IF59    Abs  967976 #EC528 -  1050  1041
%IF60    Abs  967841 #EC4A1 -   983   943
%IF61    Abs  967471 #EC32F -   842   838
%OF1     Abs  966726 #EC046 -   569   557
%OF2     Abs  966816 #EC0A0 -   598   574
%OF3     Abs  966926 #EC10E -   634   604
%OF4     Abs  966981 #EC145 -   652   640
%RP1     Abs  967771 #EC45B -   958   953
%RP11    Abs  967666 #EC3F2 -   916   884
%RP2     Abs  967816 #EC488 -   972   968
%RP3     Abs  965576 #EBBC8 -   109    89
%RP4     Abs  965746 #EBC72 -   187   167
%RP5     Abs  965856 #EBCE0 -   236   220
%RP6     Abs  966216 #EBE48 -   383   375
%RP7     Abs  967271 #EC267 -   771   739
%RP9     Abs  967486 #EC33E -   849   806
%TH46    Abs  966041 #EBFF1 -   529   432
%TH47    Abs  966641 #EBFF1 -   528   446
%TH48    Abs  966641 #EBFF1 -   527   462
%TH49    Abs  966641 #EBFF1 -   526   491
%TH50    Abs  966641 #EBFF1 -   525   503
/        Ext                -   843
2DROP    Ext                -   978  1014
>R       Ext                -   363   477   668   752   819   894  1013
ADD      Ext                -    33   905   940
AND      Ext                -    87   165   218   373   760
AT       Ext                -    28   549   563   582   615   646   688   728
                            795   873   950   965  1020
BL       Ext                -   254   337
```

```
BRNCH   Ext          -  104  182  231  381  431  445  461  490
                        502  564  593  629  647  761  766  828
                        844  906  911  956  970
C@+     Ext          -   93  171  224  379  458  474  478  501
                        515
CAT     Ext          -   84  162  212  369  438
DIGIT   Ext          -   50   96  128  174
DOCOL   Ext          -   23   44   72  122  150  200  249  332
                        358  419  540  679  714  781  859  929
                        992 1035
DROP    Ext          -   56   97  134  175  380  443  457  459
                        473  475  500  514  520  561  578  608
                        644  656  772  850  917  955  969 1045
DUP     Ext          -  250  336  423  439  451  467  496  508
                        729  796  835  874
EQUAL   Ext          -  215  255  262  270  278  286  294  302
                        310  318  338  345  454  470  481  511
                        555  572  602  618  638  691  732  736
                        756  799  803  823  877  881  901
LIT     Ext          -   48   94  101  126  172  179  213  228
                        260  268  276  284  292  300  308  316
                        343  402  452  468  479  485  509  589
                        599  616  625  635  657  689  730  734
                        754  797  801  821  839  875  879  899
                        932  980 1023
LT      Ext          -  941  951  966
MIN     Ext          -  404
MINUS   Ext          -  392  910
MULT    Ext          -   29  827
NEGAT   Ext          -  699
ONE     Ext          -  113  191  240  553
OR      Ext          -  263  271  279  287  295  303  311  319
                        346  737  765  804  882
OVER    Ext          -   83   86  161  164  211  217  259  267
                        275  283  291  299  307  362  368  372
                        390  437  554  571  601  637  733  800
                        878
R>      Ext          -  397  489  670  759  764  826  831  904
                        909  954 1015
R@      Ext          -  391
RP@     Ext          -  930  948  996
S!      Ext          -  406
SEMI    Ext          -   36   58   64  114  136  142  192  241
                        324  350  411  532  671  700  706  773
                        851  918  984 1027 1046 1052
SP@     Ext          -  963 1002
STORE   Ext          -   35   78  156  206  531  998 1004
SWAP    Ext          -  315  342  398
SWAP2   Ext          -  743  753  810  820  888  898
TIB     Ext          -  931
TWO     Ext          -  410  570
TWO/    Ext          -  393
ZBRNH   Ext          -   51   88  129  166  219  374  425  441
                        455  471  483  498  512  556  573  584
                        603  620  639  692  738  757  805  824
```



```
                             -  837  883  902  942  952  967 1021 1040
 ZEQ     Ext                 -  216  371  424  482  583  619  836
 ZERO    Ext                 -   63   76  141  154  204  430  660  979
=a+DIGI  Abs  965376 #EBB00  -   23  103  181  230
=a?ELAB  Abs  966091 #EBDCB  -  332  370
 a?PAS1  Ext                 - 1039
=a?STOK  Abs  965866 #EBCEA  -  249  497
=aCON10  Abs  965471 #EBB5F  -   72  444
=aCON16  Abs  965641 #EBC09  -  150  460
=aCONFF  Abs  965756 #EBC7C  -  200  476
=aDEPCK  Abs  967676 #EC3FC  -  929  544  683  718  785  863
=aDIG10  Abs  965416 #EBB28  -   44   85  440
=aDIG16  Abs  965586 #EBBD2  -  122  163
=aDOLAB  Abs  966141 #EBDFD  -  358  521
 aDSTK   Ext                 -  964 1003
 aERROR  Ext                 -  487 1051
=aEXPRS  Abs  967846 #EC4A6  -  992
 aIDENT  Ext                 -  405  579
 aKNOWN  Ext                 -  581
 aLC     Ext                 -  645
=aLEXSC  Abs  966281 #EBE89  -  419  609  669  697  747  814  892 1008
 aNEG1   Ext                 -   57  135
=aP2ERR  Abs  967946 #EC50A  - 1035  591  627  659  841  982 1025
 aRSTK   Ext                 -  949  997
 aSYT>   Ext                 -  580
 aTOKEN  Ext                 -  530  548  614  687  727  794  872 1019
 aVALUE  Ext                 -   27   34   77  155  205  562
=amBASE  Abs  966656 #EC000  -  540  698  705
=amBOOL  Abs  967101 #EC1BD  -  714  789  815
=amFACT  Abs  967281 #EC271  -  781  867  893
=amTERM  Abs  967496 #EC348  -  859  610 1009
=amUN-   Abs  967026 #EC172  -  679  722  748
 zLEFT   Abs     206 #000CE  - 1055 1056
 zNEXT   Abs  968192 #EC600  -    5 1055
 zSIZE   Abs    2610 #00A32  - 1054
 zTHIS   Abs  965376 #EBB00  -    4 1054
```


Input Parameters

  Source file name is GN&AS4

  Listing file name is GN/AS4::65

  Object file name is GN%AS4::65

```
                                    111111
                          0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1                      TITLE  AS5:_FORTH_ASSEMBLER_OPCODES
 2                      RDSYMB MR%GTO
 3 EC600                ABS    #EC600
 4              zTHIS   EQU    *
 5              zNEXT   EQU    #ED700
 6
 7              ************************************************
 8              ***
 9              *** MR&AS5      <840504.1037>
10              ***
11              ***
12              ***  FORTH ASSEMBLER
13              ***  ALL OPCODES OF SASM
14              ***
15              ************************************************
16              ****************************************************************
17              ***
18              ***  OP1
19              ***
20              ****************************************************************
21 EC600 0000 =aOP1   CON(5) =DOCOL          : OP1
          0
22 EC605 0000         CON(5) =aOP            OP
          0
23 EC60A 0000         CON(5) =SEMI           ;
          0
24              ****************************************************************
25              ***
26              ***  OP2
27              ***
28              ****************************************************************
29 EC60F 0000 =aOP2   CON(5) =DOCOL          : OP2
          0
30 EC614 0000         CON(5) =aOP            OP
          0
31 EC619 0000         CON(5) =FIVE+          5+
          0
32 EC61E 0000         CON(5) =SEMI           ;
          0
33              ****************************************************************
34              ***
35              ***  OP3
36              ***
37              ****************************************************************
38 EC623 0000 =aOP3   CON(5) =DOCOL          : OP3
          0
39 EC628 0000         CON(5) =aOP            OP
          0
40 EC62D 0000         CON(5) =LIT            10
          0
41 EC632 A000         CON(5) 10
          0
42 EC637 0000         CON(5) =ADD            +
          0
43 EC63C 0000         CON(5) =SEMI           ;
```



```
          0
44              ****************************************************************
45              ***
46              ***  OP4
47              ***
48              ****************************************************************
49 EC641 0000 =aOP4   CON(5) =DOCOL          : OP4
          0
50 EC646 0000         CON(5) =aOP            OP
          0
51 EC64B 0000         CON(5) =LIT            15
          0
52 EC650 F000         CON(5) 15
          0
53 EC655 0000         CON(5) =ADD            +
          0
54 EC65A 0000         CON(5) =SEMI           ;
          0
55              ****************************************************************
56              ***
57              ***  GET.FLAG
58              ***
59              *** ( NUM -- F )
60              *** ( RETURN OPFLAGS BIT SPECIFIED BY NUM )
61              ****************************************************************
62 EC65F 0000 =aGFLAG CON(5) =DOCOL          : GET.FLAG
          0
63 EC664 0000         CON(5) =aOPFLG         OPFLAGS
          0
64 EC669 0000         CON(5) =AT             @
          0
65 EC66E 0000         CON(5) =AND            AND
          0
66 EC673 0000         CON(5) =SEMI           ;
          0
67              ****************************************************************
68              ***
69              ***  LENGTH.SET
70              ***
71              *** ( -- F )
72              *** ( RETURN OPFLAGS BIT 1 )
73              ****************************************************************
74 EC678 0000 =aLENST CON(5) =DOCOL          : LENGTH.SET
          0
75 EC67D 0000         CON(5) =ONE            1
          0
76 EC682 F56C         CON(5) =aGFLAG         GET.FLAG
         E
77 EC687 0000         CON(5) =SEMI           ;           ( * F * )
          0
78              ****************************************************************
79              ***
80              ***  REQ.YES
81              ***
82              *** ( -- F )
```

```
                *** ( RETURN OPFLAGS BIT 2 )
                ********************************************************************
EC68C 00000 =aRQYES CON(5) =DOCOL          : REQ.YES
EC691 00000         CON(5) =TWO            2
EC696 F56CE         CON(5) =aGFLAG         GET.FLAG
EC69B 00000         CON(5) =SEMI           ;           ( * F * )
                ********************************************************************
                ***
                ***  IS.GOYES
                ***
                *** ( -- F )
                *** ( RETURN OPFLAGS BIT 8 )
                ********************************************************************
EC6A0 00000 =aISGYS CON(5) =DOCOL          : IS.GOYES
EC6A5 00000         CON(5) =LIT            8
EC6AA 80000         CON(5) 8
EC6AF F56CE         CON(5) =aGFLAG         GET.FLAG
EC6B4 00000         CON(5) =SEMI           ;           ( * F * )
                ********************************************************************
                ***
                ***  IS.ABS
                ***
                *** ( -- F )
                *** ( RETURN OPFLAGS BIT 16 )
                ********************************************************************
EC6B9 00000 =aISABS CON(5) =DOCOL          : IS.ABS
EC6BE 00000         CON(5) =LIT            16
EC6C3 01000         CON(5) 16
EC6C8 F56CE         CON(5) =aGFLAG         GET.FLAG
EC6CD 00000         CON(5) =SEMI           ;           ( * F * )
                ********************************************************************
                ***
                ***  IS.GSUB
                ***
                *** ( -- F )
                *** ( RETURN OPFLAGS BIT 64 )
                ********************************************************************
EC6D2 00000 =aISGSB CON(5) =DOCOL          : IS.GSUB
EC6D7 00000         CON(5) =LIT            64
```



```
EC6DC 00400         CON(5) 64
EC6E1 F56CE         CON(5) =aGFLAG         GET.FLAG
EC6E6 00000         CON(5) =SEMI           ;           ( * F * )
                ********************************************************************
                ***
                ***  FIELD.SELECT
                ***
                *** ( -- NUM )
                *** ( RETURN WHICH FIELD SELECT CHOSEN IN EXPRESSION )
                ********************************************************************
EC6EB 00000 =aFS    CON(5) =DOCOL          : FIELD.SELECT
                *
                * EXAMINE EXPRESSION FIELD FOR:
                *
EC6F0 00000         CON(5) =aEXP.F         EXPRESSION.FIELD
                *
                * "A" - ADDRESS FIELD SELECT (9)
                *
EC6F5 00000         CON(5) =2DUP           2DUP
EC6FA 00000         CON(5) =QUOTC          "
EC6FF 10            CON(2) 1
EC701 10            CON(2) 1
EC703 14            NIBASC \A\
EC705 00000         CON(5) =S=             S=
EC70A 00000         CON(5) =ZBRNH          IF
EC70F 91000         REL(5) %IF59
EC714 00000         CON(5) =LIT            9
EC719 90000         CON(5) 9
EC71E 00000         CON(5) =BRNCH
EC723 79100         REL(5) %TH59
                *
                * "W" - WORD FIELD SELECT (8)
                *
EC728                %IF59                 ELSE        ( * STR * )
EC728 00000         CON(5) =2DUP           2DUP
EC72D 00000         CON(5) =QUOTC          "
EC732 10            CON(2) 1
EC734 10            CON(2) 1
EC736 75            NIBASC \W\
```

```
   161 EC738 0000          CON(5) =S=              S=
             0
   162 EC73D 0000          CON(5) =ZBRNH           IF
             0
   163 EC742 9100          REL(5) %IF60
             0
   164 EC747 0000          CON(5) =LIT             8
             0
   165 EC74C 8000          CON(5) 8
             0
   166 EC751 0000          CON(5) =BRNCH
             0
   167 EC756 4610          REL(5) %TH60
             0
   168               *
   169               * "P" - POINTER SELECT FIELD SELECT (1)
   170               *
   171 EC75B        %IF60                          ELSE        ( * STR * )
   172 EC75B 0000          CON(5) =2DUP            2DUP
             0
   173 EC760 0000          CON(5) =QUOTC           "
             0
   174 EC765 10            CON(2) 1
   175 EC767 10            CON(2) 1
   176 EC769 05            NIBASC \P\
   177 EC76B 0000          CON(5) =S=              S=
             0
   178 EC770 0000          CON(5) =ZBRNH           IF
             0
   179 EC775 4100          REL(5) %IF65
             0
   180 EC77A 0000          CON(5) =ONE             1
             0
   181 EC77F 0000          CON(5) =BRNCH
             0
   182 EC784 6310          REL(5) %TH65
             0
   183               *
   184               * "B" - BYTE (OR EXPONENT) FIELD SELECT (7)
   185               *
   186 EC789        %IF65                          ELSE        ( * STR * )
   187 EC789 0000          CON(5) =2DUP            2DUP
             0
   188 EC78E 0000          CON(5) =QUOTC           "
             0
   189 EC793 10            CON(2) 1
   190 EC795 10            CON(2) 1
   191 EC797 24            NIBASC \B\
   192 EC799 0000          CON(5) =S=              S=
             0
   193 EC79E 0000          CON(5) =ZBRNH           IF
             0
   194 EC7A3 9100          REL(5) %IF61
             0
   195 EC7A8 0000          CON(5) =LIT             7
```



```
             0
   196 EC7AD 7000          CON(5) 7
             0
   197 EC7B2 0000          CON(5) =BRNCH
             0
   198 EC7B7 3010          REL(5) %TH61
             0
   199               *
   200               * "S" - SIGN FIELD SELECT (5)
   201               *
   202 EC7BC        %IF61                          ELSE        ( * STR * )
   203 EC7BC 0000          CON(5) =2DUP            2DUP
             0
   204 EC7C1 0000          CON(5) =QUOTC           "
             0
   205 EC7C6 10            CON(2) 1
   206 EC7C8 10            CON(2) 1
   207 EC7CA 35            NIBASC \S\
   208 EC7CC 0000          CON(5) =S=              S=
             0
   209 EC7D1 0000          CON(5) =ZBRNH           IF
             0
   210 EC7D6 9100          REL(5) %IF62
             0
   211 EC7DB 0000          CON(5) =LIT             5
             0
   212 EC7E0 5000          CON(5) 5
             0
   213 EC7E5 0000          CON(5) =BRNCH
             0
   214 EC7EA 0D00          REL(5) %TH62
             0
   215               *
   216               * "X" - EXPONENT FIELD SELECT (4)
   217               *
   218 EC7EF        %IF62                          ELSE        ( * STR * )
   219 EC7EF 0000          CON(5) =2DUP            2DUP
             0
   220 EC7F4 0000          CON(5) =QUOTC           "
             0
   221 EC7F9 10            CON(2) 1
   222 EC7FB 10            CON(2) 1
   223 EC7FD 85            NIBASC \X\
   224 EC7FF 0000          CON(5) =S=              S=
             0
   225 EC804 0000          CON(5) =ZBRNH           IF
             0
   226 EC809 9100          REL(5) %IF63
             0
   227 EC80E 0000          CON(5) =LIT             4
             0
   228 EC813 4000          CON(5) 4
             0
   229 EC818 0000          CON(5) =BRNCH
             0
```

```
230 EC81D D900          REL(5) %TH63
        0
231             *
232             * "M" - MANTISSA FIELD SELECT (6)
233             *
234 EC822        %IF63                          ELSE        ( * STR * )
235 EC822 0000          CON(5) =2DUP           2DUP
        0
236 EC827 0000          CON(5) =QUOTC          "
        0
237 EC82C 10            CON(2) 1
238 EC82E 10            CON(2) 1
239 EC830 D4            NIBASC \M\
240 EC832 0000          CON(5) =S=             S=
        0
241 EC837 0000          CON(5) =ZBRNH          IF
        0
242 EC83C 9100          REL(5) %IF64
        0
243 EC841 0000          CON(5) =LIT            6
        0
244 EC846 6000          CON(5) 6
        0
245 EC84B 0000          CON(5) =BRNCH
        0
246 EC850 A600          REL(5) %TH64
        0
247             *
248             * "XS" - EXPONENT & SIGN FIELD SELECT (3)
249             *
250 EC855        %IF64                          ELSE        ( * STR * )
251 EC855 0000          CON(5) =2DUP           2DUP
        0
252 EC85A 0000          CON(5) =QUOTC          "
        0
253 EC85F 20            CON(2) 2
254 EC861 20            CON(2) 2
255 EC863 8535          NIBASC \XS\
256 EC867 0000          CON(5) =S=             S=
        0
257 EC86C 0000          CON(5) =ZBRNH          IF
        0
258 EC871 4100          REL(5) %IF66
        0
259 EC876 0000          CON(5) =THREE          3
        0
260 EC87B 0000          CON(5) =BRNCH
        0
261 EC880 A300          REL(5) %TH66
        0
262             *
263             * "WP" - WORD POINTER FIELD SELECT (2)
264             *
265 EC885        %IF66                          ELSE        ( * STR * )
266 EC885 0000          CON(5) =2DUP           2DUP
```



```
        0
267 EC88A 0000          CON(5) =QUOTC          "
        0
268 EC88F 20            CON(2) 2
269 EC891 20            CON(2) 2
270 EC893 7505          NIBASC \WP\
271 EC897 0000          CON(5) =S=             S=
        0
272 EC89C 0000          CON(5) =ZBRNH          IF
        0
273 EC8A1 4100          REL(5) %IF67
        0
274 EC8A6 0000          CON(5) =TWO            2
        0
275 EC8AB 0000          CON(5) =BRNCH
        0
276 EC8B0 A000          REL(5) %TH67
        0
277             *
278             * NONE OF THE ABOVE
279             *
280 EC8B5        %IF67                          ELSE        ( * STR * )
281 EC8B5 0000          CON(5) =ZERO           0
        0
282 EC8BA        %TH67                          THEN
283 EC8BA        %TH66                          THEN
284 EC8BA        %TH65                          THEN
285 EC8BA        %TH64                          THEN
286 EC8BA        %TH63                          THEN
287 EC8BA        %TH62                          THEN
288 EC8BA        %TH61                          THEN
289 EC8BA        %TH60                          THEN
290 EC8BA        %TH59                          THEN        ( * STR NUM * )
291             *
292             * DROP STRING FROM STACK
293             *
294 EC8BA 0000          CON(5) =>R             >R
        0
295 EC8BF 0000          CON(5) =2DROP          2DROP
        0
296 EC8C4 0000          CON(5) =R>             R>
        0
297 EC8C9 0000          CON(5) =SEMI           ;           ( * NUM * )
        0
298             ********************************************************************
299             ***
300             ***  EXPR
301             ***
302             *** ( -- NUM )
303             *** ( EVALUATE EXPRESSION.FIELD )
304             ********************************************************************
305 EC8CE 0000 =aEXPR  CON(5) =DOCOL          : EXPR
        0
306 EC8D3 0000          CON(5) =aEXP.F         EXPRESSION.FIELD
        0
```

```
 307 EC8D8 0000         CON(5) =aEXPRS        EXPRESSION
         0
 308 EC8DD 0000         CON(5) =SEMI          ;           ( * NUM * )
         0
 309           ****************************************************************
 310           ***
 311           ***  0<=EXP<=15
 312           ***
 313           *** ( -- NUM F )
 314           *** ( EVALUATE EXPRESSION.FIELD BETWEEN 0 AND 15 )
 315           ****************************************************************
 316 EC8E2 0000 =aEXP15 CON(5) =DOCOL        : 0<=EXP<=15
         0
 317           *
 318           * EVALUATE EXPRESSION FIELD
 319           *
 320 EC8E7 EC8C         CON(5) =aEXPR         EXPR
         E
 321 EC8EC 0000         CON(5) =DUP           DUP         ( * NUM NUM * )
         0
 322           *
 323           * NUM >= 0 ?
 324           *
 325 EC8F1 0000         CON(5) =aNEG1         -1
         0
 326 EC8F6 0000         CON(5) =GT            >
         0
 327 EC8FB 0000         CON(5) =OVER          OVER
         0
 328           *
 329           * NUM <= 15 ?
 330           *
 331 EC900 0000         CON(5) =LIT           16
         0
 332 EC905 0100         CON(5) 16
         0
 333 EC90A 0000         CON(5) =LT            <
         0
 334           *
 335           * AND
 336           *
 337 EC90F 0000         CON(5) =AND           AND
         0
 338 EC914 0000         CON(5) =SEMI          ;           ( * NUM F * )
         0
 339           ****************************************************************
 340           ***
 341           ***  1<=EXP<=16
 342           ***
 343           *** ( -- NUM F )
 344           *** ( EVALUATE EXPRESSION.FIELD BETWEEN 1 AND 16 )
 345           ****************************************************************
 346 EC919 0000 =aEXP16 CON(5) =DOCOL        : 1<=EXP<=16
         0
 347           *
```



```
 348           * EVALUATE EXPRESSION FIELD
 349           *
 350 EC91E EC8C         CON(5) =aEXPR         EXPR
         E
 351           *
 352           * NUM > 0
 353           *
 354 EC923 0000         CON(5) =DUP           DUP         ( * NUM NUM * )
         0
 355 EC928 0000         CON(5) =GTZ           0>
         0
 356 EC92D 0000         CON(5) =OVER          OVER
         0
 357           *
 358           * NUM <= 16
 359           *
 360 EC932 0000         CON(5) =LIT           17
         0
 361 EC937 1100         CON(5) 17
         0
 362 EC93C 0000         CON(5) =LT            <
         0
 363           *
 364           * AND
 365           *
 366 EC941 0000         CON(5) =AND           AND
         0
 367 EC946 0000         CON(5) =SEMI          ;           ( * NUM F * )
         0
 368           ****************************************************************
 369           ***
 370           ***  RANGE
 371           ***
 372           *** ( DEST CURR NIBS -- OFFSET FLAG )
 373           *** ( ENSURE THE DIFFERENCE BETWEEN DEST AND CURR FITS IN
 374           ***   A SIGNED NUMBER OF LENGTH NIBS.  RETURN THE DIFFERENCE
 375           ***   DEST-CURR & A FLAG.
 376           ***
 377           ****************************************************************
 378 EC94B 059C =aRANGE CON(5) $RANGE         primitive RANGE
         E
 379 EC950      $RANGE
 380           *
 381           * GET NUMBER OF NIBBLES INTO P
 382           *
 383 EC950 147          C=DAT1 A
 384 EC953 174          D1=D1+ 5              POP FIRST ELEMENT
 385 EC956 80D0         P=C    0
 386 EC95A 0D           P=P-1
 387           *
 388           * GET CURRENT LOCATION & DESTINATION LOCATION
 389           *
 390 EC95C AF2          C=0    W
 391 EC95F 147          C=DAT1 A
 392 EC962 174          D1=D1+ 5
```

```
393 EC965 AF7          D=C    W
394 EC968 147          C=DAT1 A
395 EC96B D0           A=0    A              INITIALIZE RETURN FLAG
396             *
397             * RETURN OFFSET TO STACK & MAKE OUR OFFSET POSITIVE
398             *
399 EC96D B7B          C=C-D  W
400 EC970 145          DAT1=C A
401 EC973 550          GONC   POS            POSITIVE RESULT
402 EC976 BFE          C=-C-1 W              -1 IS FOR LARGER NEGATIVE RANGE
403             *
404             * CHECK IF DOUBLING NUMBER CREATES OVERFLOW.
405             *
406 EC979 A16  POS      C=C+C  WP
407 EC97C 411          GOC    FAIL
408             *
409             * SHIFT NUMBER OUT OF C REGISTER
410             *
411 EC97F BF6  SHIFT    CSR    W
412 EC982 0D           P=P-1
413 EC984 5AF          GONC   SHIFT
414             *
415             * SIGN FIELD LEFT IN C.  ENSURE SIGN IS ZERO
416             *
417 EC987 CE           C=C-1  A
418 EC989 540          GONC   FAIL
419             *
420             * RETURN FLAG
421             *
422 EC98C CC   SUCC     A=A-1  A              TRUE FLAG
423 EC98E 1C4  FAIL     D1=D1- 5              PUSH FLAG TO STACK
424 EC991 141          DAT1=A A
425 EC994 03           RTNCC
426             ****************************************************************
427             ***
428             ***  FILL.OP
429             ***
430             *** ( NUM PTR CNT -- )
431             *** ( PUT CNT NIBBLES OF NUM INTO OP[PTR] )
432             ****************************************************************
433 EC996 0000 =aFILOP CON(5) =DOCOL          : FILL.OP
        0
434 EC99B 0000         CON(5) =OVER           OVER
        0
435 EC9A0 0000         CON(5) =ADD            +
        0
436 EC9A5 0000         CON(5) =SWAP           SWAP
        0
437 EC9AA 0000         CON(5) =XDO            DO          ( * NUM * )
        0
438             *
439             * LOOP THROUGH EACH OP[PTR]
440             *
441 EC9AF       %DO7
442             *
```



```
443             * EXTRACT DIGIT
444             *
445 EC9AF 0000         CON(5) =DUP            DUP
        0
446 EC9B4 0000         CON(5) =LTZ            0<
        0
447 EC9B9 0000         CON(5) =LIT            16
        0
448 EC9BE 0100         CON(5) 16
        0
449 EC9C3 0000         CON(5) =UDIV           UM/MOD
        0
450             *
451             * STORE IN APPROPRIATE OP[PTR]
452             *
453 EC9C8 0000         CON(5) =SWAP           SWAP        ( * QUO REM * )
        0
454 EC9CD 0000         CON(5) =R@             I
        0
455 EC9D2 0000         CON(5) =a>OPx          >OPx
        0
456 EC9D7 0000         CON(5) =STORE          !           ( * QUO * )
        0
457 EC9DC 0000         CON(5) =XLOOP          LOOP
        0
458 EC9E1 ECFF         REL(5) %DO7
        F
459             *
460             * RETURN NOTHING
461             *
462 EC9E6 0000         CON(5) =DROP           DROP
        0
463 EC9EB 0000         CON(5) =SEMI           ;           ( * * )
        0
464             ****************************************************************
465             ***
466             ***  ?PASS=1
467             ***
468             *** ( -- F )
469             *** ( RETURN FLAG TRUE IF PASS ONE )
470             ****************************************************************
471 EC9F0 0000 =a?PAS1 CON(5) =DOCOL          : ?PASS=1
        0
472 EC9F5 0000         CON(5) =aPASS          PASS
        0
473 EC9FA 0000         CON(5) =AT             @
        0
474 EC9FF 0000         CON(5) =ONE            1
        0
475 ECA04 0000         CON(5) =EQUAL          =
        0
476 ECA09 0000         CON(5) =SEMI           ;           ( * F * )
        0
477             ****************************************************************
478             ***
```

```
479                *** REGTEST
480                ***
481                *** ( -- )
482                *** ( REGISTER TEST OPCODES )
483                ********************************************************************
484 ECA0E 00000 =aREGTE CON(5) =DOCOL          : REGTEST
485                *
486                * FIELD SELECT
487                *
488 ECA13 BE6CE         CON(5) =aFS            FIELD.SELECT
489                *
490                * VALID FIELD SELECT ?
491                *
492 ECA18 00000         CON(5) =?DUP           ?DUP
493 ECA1D 00000         CON(5) =ZBRNH          IF
494 ECA22 B9000         REL(5) %IF68
495                *
496                * YES.
497                *
498 ECA27 00000         CON(5) =DUP            DUP          ( * NUM NUM * )
499 ECA2C 00000         CON(5) =LIT            9
500 ECA31 90000         CON(5) 9
501 ECA36 00000         CON(5) =EQUAL          =
502 ECA3B 00000         CON(5) =ZBRNH          IF           ( * NUM * )
503 ECA40 32000         REL(5) %IF69
504                *
505                * FIELD SELECT "A"
506                *
507 ECA45 00000         CON(5) =aREGRP         REG.GROUP
508 ECA4A 00000         CON(5) =AT             @
509 ECA4F 00000         CON(5) =ADD            +
510 ECA54 F06CE         CON(5) =aOP2           OP2
511 ECA59 00000         CON(5) =STORE          !            ( * * )
512 ECA5E 00000         CON(5) =SEMI           ;            ( * * )
513                *
514                * ALL OTHER FIELD SELECTS
515                *
516 ECA63        %IF69                         ELSE         ( * NUM * )
```



```
517 ECA63 00000         CON(5) =LIT            9
518 ECA68 90000         CON(5) 9
519 ECA6D 006CE         CON(5) =aOP1           OP1
520 ECA72 00000         CON(5) =STORE          !
521 ECA77 00000         CON(5) =ONE-           1-
522 ECA7C F06CE         CON(5) =aOP2           OP2
523 ECA81 00000         CON(5) =STORE          !            ( * * )
524                *
525                * IF REGISTER GROUP 2, THEN OP2 = OP2 + 8
526                *
527 ECA86 00000         CON(5) =aREGRP         REG.GROUP
528 ECA8B 00000         CON(5) =AT             @
529 ECA90 00000         CON(5) =TWO            2
530 ECA95 00000         CON(5) =EQUAL          =
531 ECA9A 00000         CON(5) =ZBRNH          IF
532 ECA9F 91000         REL(5) %IF70
533 ECAA4 00000         CON(5) =LIT            8
534 ECAA9 80000         CON(5) 8
535 ECAAE F06CE         CON(5) =aOP2           OP2
536 ECAB3 00000         CON(5) =PSTOR          +!
537 ECAB8        %IF70                         THEN         ( * * )
538 ECAB8 00000         CON(5) =SEMI           ;            ( * * )
539                *
540                * INVALID FIELD SELECT
541                *
542 ECABD        %IF68
543 ECABD 00000         CON(5) =LIT            27
544 ECAC2 B1000         CON(5) 27              'illegal word select'
545 ECAC7 00000         CON(5) =aERROR         ERROR
546 ECACC 00000         CON(5) =SEMI           ;            ( * * )
547                ********************************************************************
548                ***
549                *** REGARITH
```

```
550             ***
551             *** ( -- [NUM] )
552             *** ( REGISTER ARITHMETIC OPCODES )
553             ****************************************************************
554 ECAD1 0000 =aREGAR CON(5) =DOCOL          : REGARITH
        0
555 ECAD6 0F9C          CON(5) =a?PAS1        ?PASS=1
        E
556 ECADB 0000          CON(5) =ZBRNH         IF          ( * * )
        0
557 ECAE0 7300          REL(5) %IF71
        0
558             *
559             * PASS 1 HANDLER
560             *
561 ECAE5 BE6C          CON(5) =aFS           FIELD.SELECT
        E
562 ECAEA 0000          CON(5) =LIT           9
        0
563 ECAEF 9000          CON(5) 9
        0
564 ECAF4 0000          CON(5) =EQUAL         =           ( * F * )
        0
565 ECAF9 0000          CON(5) =ZBRNH         IF
        0
566 ECAFE F000          REL(5) %IF72
        0
567             *
568             * FIELD SELECT "A" - RETURN LENGTH 2
569             *
570 ECB03 0000          CON(5) =TWO           2
        0
571 ECB08 0000          CON(5) =SEMI          ;           ( * NUM * )
        0
572             *
573             * ALL OTHER FIELD SELECT - RETURN LENGTH 3
574             *
575 ECB0D        %IF72                        ELSE        ( * * )
576 ECB0D 0000          CON(5) =THREE         3
        0
577 ECB12 0000          CON(5) =SEMI          ;           ( * NUM * )
        0
578             *
579             * PASS 2 HANDLER
580             *
581 ECB17        %IF71                        ELSE        ( * * )
582 ECB17 BE6C          CON(5) =aFS           FIELD.SELECT
        E
583             *
584             * VALID FIELD SELECT ?
585             *
586 ECB1C 0000          CON(5) =?DUP          ?DUP
        0
587 ECB21 0000          CON(5) =ZBRNH         IF
        0
```



```
588 ECB26 6E00          REL(5) %IF73
        0
589 ECB2B 0000          CON(5) =DUP           DUP         ( * NUM NUM * )
        0
590 ECB30 0000          CON(5) =LIT           9
        0
591 ECB35 9000          CON(5) 9
        0
592 ECB3A 0000          CON(5) =EQUAL         =
        0
593 ECB3F 0000          CON(5) =ZBRNH         IF          ( * NUM * )
        0
594 ECB44 5500          REL(5) %IF74
        0
595             *
596             * FIELD SELECT "A"
597             *   OP2 = OP3
598             *   OP1 = 11 + ARITHMETIC GROUP
599             *
600 ECB49 0000          CON(5) =DROP          DROP
        0
601 ECB4E 0000          CON(5) =TWO           2
        0
602 ECB53 0000          CON(5) =aOPLEN        OPLEN
        0
603 ECB58 0000          CON(5) =STORE         !           ( * * )
        0
604 ECB5D 326C          CON(5) =aOP3          OP3
        E
605 ECB62 0000          CON(5) =AT            @
        0
606 ECB67 F06C          CON(5) =aOP2          OP2
        E
607 ECB6C 0000          CON(5) =STORE         !           ( * * )
        0
608 ECB71 0000          CON(5) =aARGRP        ARITH.GROUP
        0
609 ECB76 0000          CON(5) =AT            @           ( * NUM * )
        0
610 ECB7B 0000          CON(5) =LIT           11
        0
611 ECB80 B000          CON(5) 11
        0
612 ECB85 0000          CON(5) =ADD           +
        0
613 ECB8A 006C          CON(5) =aOP1          OP1
        E
614 ECB8F 0000          CON(5) =STORE         !
        0
615 ECB94 0000          CON(5) =SEMI          ;           ( * * )
        0
616             *
617             * ALL OTHER FIELD SELECTS
618             *   OP1 = 10 (IF REGISTER GROUP C OR D)
619             *   OP1 = 11 (IF REGISTER GROUP E OR F)
```

```
620            *   OP2 = <FIELD SELECT> - 1 + 0 (IF REGISTER GROUP C OR E)
621            *   OP2 = <FILED SELECT> - 1 + 8 (IF REGISTER GROUP D OR F)
622            *
623 ECB99       %IF74                       ELSE        ( * NUM * )
624 ECB99 0000          CON(5) =ONE-        1-
          0
625 ECB9E F06C          CON(5) =aOP2        OP2
          E
626 ECBA3 0000          CON(5) =STORE       !           ( * * )
          0
627 ECBA8 0000          CON(5) =aARGRP      ARITH.GROUP
          0
628 ECBAD 0000          CON(5) =AT          @
          0
629 ECBB2 0000          CON(5) =DUP         DUP         ( * NUM NUM * )
          0
630 ECBB7 0000          CON(5) =ONE-        1-
          0
631 ECBBC 0000          CON(5) =TWO/        2/
          0
632 ECBC1 0000          CON(5) =LIT         10
          0
633 ECBC6 A000          CON(5) 10
          0
634 ECBCB 0000          CON(5) =ADD         +
          0
635 ECBD0 006C          CON(5) =aOP1        OP1
          E
636 ECBD5 0000          CON(5) =STORE       !           ( * NUM * )
          0
637 ECBDA 0000          CON(5) =DUP         DUP
          0
638 ECBDF 0000          CON(5) =TWO/        2/
          0
639 ECBE4 0000          CON(5) =TWO*        2*
          0
640 ECBE9 0000          CON(5) =EQUAL       =           ( * F * )
          0
641 ECBEE 0000          CON(5) =LIT         -8
          0
642 ECBF3 8FFF          CON(5) 0-8
          F
643 ECBF8 0000          CON(5) =MULT        *
          0
644 ECBFD F06C          CON(5) =aOP2        OP2
          E
645 ECC02 0000          CON(5) =PSTOR       +!
          0
646 ECC07 0000          CON(5) =SEMI        ;           ( * * )
          0
647            *
648            * INVALID FIELD SELECT
649            *
650 ECC0C       %IF73                       ELSE        ( * * )
651 ECC0C 0000          CON(5) =LIT         27
```



```
          0
652 ECC11 B100          CON(5) 27           'illegal word select'
          0
653 ECC16 0000          CON(5) =aERROR      ERROR
          0
654 ECC1B 0000          CON(5) =SEMI        ;           ( * * )
          0
655            ***************************************************************
656            ***
657            ***  REGLOGIC
658            ***
659            *** ( -- )
660            *** ( REGISTER LOGIC OPCODES )
661            ***************************************************************
662 ECC20 0000 =aREGLO CON(5) =DOCOL        : REGLOGIC
          0
663 ECC25 BE6C          CON(5) =aFS         FIELD.SELECT
          E
664            *
665            * VALID FIELD SELECT ?
666            *
667 ECC2A 0000          CON(5) =?DUP        ?DUP
          0
668 ECC2F 0000          CON(5) =ZBRNH       IF
          0
669 ECC34 6400          REL(5) %IF75
          0
670 ECC39 0000          CON(5) =DUP         DUP         ( * NUM NUM * )
          0
671 ECC3E 0000          CON(5) =LIT         9
          0
672 ECC43 9000          CON(5) 9
          0
673 ECC48 0000          CON(5) =EQUAL       =
          0
674 ECC4D 0000          CON(5) =ZBRNH       IF          ( * NUM * )
          0
675 ECC52 4100          REL(5) %IF76
          0
676            *
677            * FIELD SELECT A
678            *   OP3 = 15
679            *
680 ECC57 0000          CON(5) =LIT         7
          0
681 ECC5C 7000          CON(5) 7
          0
682 ECC61 0000          CON(5) =ADD         +
          0
683            *
684            * ALL OTHER FIELD SELECTS
685            *   OP3 = <FIELD SELECT> - 1
686            *
687 ECC66       %IF76                       THEN        ( * NUM * )
688 ECC66 0000          CON(5) =ONE-        1-
```

```
          0
689 ECC6B 326C          CON(5) =aOP3            OP3
          E
690 ECC70 0000          CON(5) =STORE           !
          0
691 ECC75 0000          CON(5) =SEMI            ;           ( * * )
          0
692               *
693               * INVALID FIELD SELECT
694               *
695 ECC7A          %IF75
696 ECC7A 0000          CON(5) =LIT             27
          0
697 ECC7F B100          CON(5) 27               'illegal word select'
          0
698 ECC84 0000          CON(5) =aERROR          ERROR
          0
699 ECC89 0000          CON(5) =SEMI            ;           ( * * )
          0
700               ****************************************************************
701               ***
702               ***  BRANCHES
703               ***
704               *** ( -- )
705               *** ( ABSOLUTE AND RELATIVE BRANCHES )
706               ****************************************************************
707 ECC8E 0000 -aBRANC CON(5) =DOCOL           : BRANCHES
          0
708               *
709               * CHECK IF WE'RE A GOYES STATEMENT AND THE PREVIOUS
710               * LINE WAS A TEST INSTRUCTION
711               *
712 ECC93 0000          CON(5) =aLSTRQ          LAST.REQ
          0
713 ECC98 0000          CON(5) =AT              @
          0
714 ECC9D 0000          CON(5) =ZEQ             0=          ( * F * )
          0
715 ECCA2 0A6C          CON(5) =aISGYS          IS.GOYES
          E
716 ECCA7 0000          CON(5) =AND             AND
          0
717 ECCAC 0000          CON(5) =ZBRNH           IF          ( * * )
          0
718 ECCB1 4100          REL(5) %IF77
          0
719               *
720               * CHECK FAILS
721               *
722 ECCB6 0000          CON(5) =LIT             29
          0
723 ECCBB D100          CON(5) 29               'needs previous test instruction'
          0
724 ECCC0 0000          CON(5) =aERROR          ERROR       ( * * )
          0
```



```
725               *
726               * EVALUATE EXPRESSION FIELD
727               *
728 ECCC5          %IF77                   THEN
729 ECCC5 EC8C          CON(5) =aEXPR           EXPR
          E
730               *
731               * CHECK IF BRANCH IS RELATIVE
732               *
733 ECCCA 9B6C          CON(5) =aISABS          IS.ABS
          E
734 ECCCF 0000          CON(5) =ZEQ             0=
          0
735 ECCD4 0000          CON(5) =ZBRNH           IF          ( * NUM * )
          0
736 ECCD9 F500          REL(5) %IF78
          0
737               *
738               * RELATIVE BRANCH
739               *
740 ECCDE 0000          CON(5) =aLC             LC
          0
741 ECCE3 0000          CON(5) =AT              @
          0
742               *
743               * GOSUBS REQUIRE RELATIVE BRANCH FROM END OF INSTRUCTION
744               *
745 ECCE8 2D6C          CON(5) =aISGSB          IS.GSUB
          E
746 ECCED 0000          CON(5) =ZBRNH           IF          ( * NUM NUM' * )
          0
747 ECCF2 9100          REL(5) %IF79
          0
748 ECCF7 0000          CON(5) =aOPLEN          OPLEN
          0
749 ECCFC 0000          CON(5) =AT              @           ( * NUM NUM' NUM" * )
          0
750 ECD01 0000          CON(5) =BRNCH           ELSE        ( * NUM NUM' * )
          0
751 ECD06 4100          REL(5) %TH79
          0
752               *
753               * OTHER BRANCHES REQUIRE RELATIVE BRANCH FROM BEGINNING OF OFFSET
754               * FIELD.
755               *
756 ECD0B          %IF79
757 ECD0B 0000          CON(5) =aFILMK          FILL.MARK
          0
758 ECD10 0000          CON(5) =AT              @
          0
759 ECD15 0000          CON(5) =ONE-            1-          ( * NUM NUM' NUM" * )
          0
760               *
761               * CALCULATE RELATIVE OFFSET
762               *
```

```
763 ECD1A       %TH79                           THEN
764 ECD1A 0000          CON(5) =ADD             +
          0
765             *
766             * ENSURE IN RANGE
767             *
768 ECD1F 0000          CON(5) =aFILEN          FILL.LENGTH
          0
769 ECD24 0000          CON(5) =AT              @           ( * DEST CURR NIBS * )
          0
770 ECD29 B49C          CON(5) =aRANGE          RANGE       ( * OFF F * )
          E
771 ECD2E 0000          CON(5) =ZBRNH           IF          ( * NUM * )
          0
772 ECD33 3200          REL(5) %IF80
          0
773             *
774             * FILL OPCODES CORRESPONDING TO OFFSET FIELD
775             *
776 ECD38        %IF78                                      ( * EXPR * )
777 ECD38 0000          CON(5) =aFILMK          FILL.MARK
          0
778 ECD3D 0000          CON(5) =AT              @           ( * NUM PTR * )
          0
779 ECD42 0000          CON(5) =aFILEN          FILL.LENGTH
          0
780 ECD47 0000          CON(5) =AT              @           ( * NUM PTR CNT * )
          0
781 ECD4C 699C          CON(5) =aFILOP          FILL.OP     ( * * )
          E
782 ECD51 0000          CON(5) =SEMI            ;           ( * * )
          0
783             *
784             * OFFSET TOO LARGE FOR OFFSET FIELD
785             *
786 ECD56        %IF80                          ELSE        ( * NUM * )
787 ECD56 0000          CON(5) =DROP            DROP
          0
788 ECD5B 0000          CON(5) =LIT             28
          0
789 ECD60 C100          CON(5) 28               'jump or value too large'
          0
790 ECD65 0000          CON(5) =aERROR          ERROR
          0
791 ECD6A 0000          CON(5) =SEMI            ;           ( * * )
          0
792             ****************************************************************
793             ***
794             ***  RTNYES
795             ***
796             *** ( -- )
797             *** ( RTNYES OPCODE )
798             ****************************************************************
799 ECD6F 0000 =aRTNYS CON(5) =DOCOL           : RTNYES
          0
```



```
800             *
801             * SINCE WE'RE A RTNYES INSTRUCTION, ALL WE HAVE TO DO IS CHECK
802             * IF THE PREVIOUS LINE WAS A TEST INSTRUCTION
803             *
804 ECD74 0000          CON(5) =aLSTRQ          LAST.REQ
          0
805 ECD79 0000          CON(5) =AT              @
          0
806 ECD7E 0000          CON(5) =ZEQ             0=
          0
807 ECD83 0000          CON(5) =ZBRNH           IF          ( * * )
          0
808 ECD88 4100          REL(5) %IF81
          0
809             *
810             * CHECK FAILED
811             *
812 ECD8D 0000          CON(5) =LIT             29
          0
813 ECD92 D100          CON(5) 29               'needs previous test instruction'
          0
814 ECD97 0000          CON(5) =aERROR          ERROR
          0
815             *
816             * ALL DONE.
817             *
818 ECD9C        %IF81                          THEN
819 ECD9C 0000          CON(5) =SEMI            ;           ( * * )
          0
820             ****************************************************************
821             ***
822             ***  PTRTEST
823             ***
824             *** ( -- )
825             *** ( POINTER TEST OPCODES )
826             ****************************************************************
827 ECDA1 0000 =aPTRTE CON(5) =DOCOL           : PTRTEST
          0
828             *
829             * EVALUATE EXPRESSION FIELD
830             *
831 ECDA6 2E8C          CON(5) =aEXP15          0<=EXP<=15
          E
832 ECDAB 0000          CON(5) =ZBRNH           IF          ( * NUM * )
          0
833 ECDB0 4100          REL(5) %IF82
          0
834             *
835             * EXPRESSION IN RANGE [0,15]
836             *
837 ECDB5 326C          CON(5) =aOP3            OP3
          E
838 ECDBA 0000          CON(5) =STORE           !
          0
839 ECDBF 0000          CON(5) =SEMI            ;           ( * * )
```

```
          0
   840            *
   841            * EXPRESSION OUT OF RANGE
   842            *
   843 ECDC4       %IF82                       ELSE        ( * NUM * )
   844 ECDC4 0000          CON(5) =DROP         DROP
          0
   845 ECDC9 0000          CON(5) =LIT          30
          0
   846 ECDCE E100          CON(5) 30            'illegal pointer position'
          0
   847 ECDD3 0000          CON(5) =aERROR       ERROR
          0
   848 ECDD8 0000          CON(5) =SEMI         ;           ( * * )
          0
   849            ****************************************************************
   850            ***
   851            ***  STATTEST
   852            ***
   853            *** ( -- )
   854            *** ( STATUS TEST OPCODES )
   855            ****************************************************************
   856 ECDDD 0000 =aSTATT CON(5) =DOCOL        : STATTEST
          0
   857            *
   858            * EVALUATE EXPRESSION
   859            *
   860 ECDE2 2E8C          CON(5) =aEXP15       0<=EXP<=15
          E
   861 ECDE7 0000          CON(5) =ZBRNH        IF
          0
   862 ECDEC 4100          REL(5) %IF83
          0
   863            *
   864            * EXPRESSION IN RANGE [0,15]
   865            *
   866 ECDF1 326C          CON(5) =aOP3         OP3
          E
   867 ECDF6 0000          CON(5) =STORE        !
          0
   868 ECDFB 0000          CON(5) =SEMI         ;           ( * * )
          0
   869            *
   870            * EXPRESSION OUT OF RANGE
   871            *
   872 ECE00       %IF83                       ELSE        ( * NUM * )
   873 ECE00 0000          CON(5) =DROP         DROP
          0
   874 ECE05 0000          CON(5) =LIT          31
          0
   875 ECE0A F100          CON(5) 31            'illegal status bit'
          0
   876 ECE0F 0000          CON(5) =aERROR       ERROR
          0
   877 ECE14 0000          CON(5) =SEMI         ;           ( * * )
```



```
          0
   878            ****************************************************************
   879            ***
   880            ***  SETPTR
   881            ***
   882            *** ( -- )
   883            *** ( SET POINTER OPCODES )
   884            ****************************************************************
   885 ECE19 0000 =aSETPT CON(5) =DOCOL        : SETPTR
          0
   886            *
   887            * EVALUATE EXPRESSION
   888            *
   889 ECE1E 2E8C          CON(5) =aEXP15       0<=EXP<=15
          E
   890 ECE23 0000          CON(5) =ZBRNH        IF          ( * NUM * )
          0
   891 ECE28 E100          REL(5) %IF84
          0
   892            *
   893            * EXPRESSION IN RANGE [0,15].  STORE EXPRESSION NIBBLE AT END
   894            * OF INSTRUCTION.
   895            *
   896 ECE2D 0000          CON(5) =aOPLEN       OPLEN
          0
   897 ECE32 0000          CON(5) =AT           @
          0
   898 ECE37 0000          CON(5) =a>OPx        >OPx
          0
   899 ECE3C 0000          CON(5) =STORE        !
          0
   900 ECE41 0000          CON(5) =SEMI         ;           ( * * )
          0
   901            *
   902            * EXPRESSION OUT OF RANGE
   903            *
   904 ECE46       %IF84                       ELSE        ( * NUM * )
   905 ECE46 0000          CON(5) =DROP         DROP
          0
   906 ECE4B 0000          CON(5) =LIT          30
          0
   907 ECE50 E100          CON(5) 30            'illegal pointer position'
          0
   908 ECE55 0000          CON(5) =aERROR       ERROR
          0
   909 ECE5A 0000          CON(5) =SEMI         ;           ( * * )
          0
   910            ****************************************************************
   911            ***
   912            ***  SETSTAT
   913            ***
   914            *** ( -- )
   915            *** ( SET STATUS OPCODES )
   916            ****************************************************************
   917 ECE5F 0000 =aSETST CON(5) =DOCOL        : SETSTAT
```

```
         0
  918            *
  919            * EVALUATE EXPRESSION
  920            *
  921 ECE64 2E8C          CON(5) =aEXP15          0<=EXP<=15
         E
  922 ECE69 0000          CON(5) =ZBRNH           IF          ( * NUM * )
         0
  923 ECE6E 4100          REL(5) %IF85
         0
  924            *
  925            * EXPRESSION IN RANGE [0,15]
  926            *
  927 ECE73 326C          CON(5) =aOP3            OP3
         E
  928 ECE78 0000          CON(5) =STORE           !
         0
  929 ECE7D 0000          CON(5) =SEMI            ;           ( * * )
         0
  930            *
  931            * EXPRESSION OUT OF RANGE
  932            *
  933 ECE82        %IF85                          ELSE        ( * NUM * )
  934 ECE82 0000          CON(5) =LIT             31
         0
  935 ECE87 F100          CON(5) 31               'illegal status bit'
         0
  936 ECE8C 0000          CON(5) =aERROR          ERROR
         0
  937 ECE91 0000          CON(5) =DROP            DROP
         0
  938 ECE96 0000          CON(5) =SEMI            ;           ( * * )
         0
  939            ****************************************************************
  940            ***
  941            ***  DPARITH
  942            ***
  943            *** ( -- )
  944            *** ( DATA POINTER ARITHMETIC OPCODES )
  945            ****************************************************************
  946 ECE9B 0000 =aDPARI CON(5) =DOCOL           : DPARITH
         0
  947            *
  948            * EVALUATE EXPRESSION
  949            *
  950 ECEA0 919C          CON(5) =aEXP16          1<=EXP<=16
         E
  951 ECEA5 0000          CON(5) =ZBRNH           IF
         0
  952 ECEAA 9100          REL(5) %IF86
         0
  953            *
  954            * EXPRESSION IN RANGE [1,16]
  955            *
  956 ECEAF 0000          CON(5) =ONE-            1-
```



```
         0
  957 ECEB4 326C          CON(5) =aOP3            OP3
         E
  958 ECEB9 0000          CON(5) =STORE           !
         0
  959 ECEBE 0000          CON(5) =SEMI            ;           ( * * )
         0
  960            *
  961            * EXPRESSION OUT OF RANGE
  962            *
  963 ECEC3        %IF86                          ELSE        ( * NUM * )
  964 ECEC3 0000          CON(5) =LIT             32
         0
  965 ECEC8 0200          CON(5) 32               'illegal dp arith value'
         0
  966 ECECD 0000          CON(5) =aERROR          ERROR
         0
  967 ECED2 0000          CON(5) =DROP            DROP
         0
  968 ECED7 0000          CON(5) =SEMI            ;           ( * * )
         0
  969            ****************************************************************
  970            ***
  971            ***  DATATRANS
  972            ***
  973            *** ( -- [NUM] )
  974            *** ( DATA TRANSFER OPCODES )
  975            ****************************************************************
  976 ECEDC 0000 =aDATAT CON(5) =DOCOL           : DATATRANS
         0
  977            *
  978            * FIELD SELECT
  979            *
  980 ECEE1 BE6C          CON(5) =aFS             FIELD.SELEC ( * NUM * )
         E
  981 ECEE6 0F9C          CON(5) =a?PAS1          ?PASS=1
         E
  982 ECEEB 0000          CON(5) =ZBRNH           IF          ( * NUM * )
         0
  983 ECEF0 5500          REL(5) %IF87
         0
  984            *
  985            * PASS 1 PROCESSING.
  986            *
  987 ECEF5 0000          CON(5) =DUP             DUP
         0
  988 ECEFA 0000          CON(5) =LIT             FS 'B'
         0
  989 ECEFF 7000          CON(5) 7
         0
  990 ECF04 0000          CON(5) =EQUAL           =
         0
  991 ECF09 0000          CON(5) =SWAP            SWAP
         0
  992 ECF0E 0000          CON(5) =LIT             FS 'A'
```

```
          0
  993 ECF13 9000          CON(5) 9
          0
  994 ECF18 0000          CON(5) =EQUAL         =
          0
  995 ECF1D 0000          CON(5) =OR            OR          ( * F * )
          0
  996 ECF22 0000          CON(5) =ZBRNH         IF
          0
  997 ECF27 F000          REL(5) %IF88
          0
  998             *
  999             * IF FIELD SELECT IS "A" OR "B" THEN RETURN LENGTH 3
 1000             *
 1001 ECF2C 0000          CON(5) =THREE         3
          0
 1002 ECF31 0000          CON(5) =SEMI          ;           ( * NUM * )
          0
 1003             *
 1004             * OTHER FIELD SELECTS RETURN LEGNTH 4
 1005             *
 1006 ECF36        %IF88
 1007 ECF36 0000          CON(5) =LIT           4
          0
 1008 ECF3B 4000          CON(5) 4
          0
 1009 ECF40 0000          CON(5) =SEMI          ;           ( * NUM * )
          0
 1010             *
 1011             * PASS 2 PROCESSING.
 1012             *
 1013 ECF45        %IF87                        ELSE        ( * NUM * )
 1014             *
 1015             * VALID FIELD SELECT ?
 1016             *
 1017 ECF45 0000          CON(5) =?DUP          ?DUP
          0
 1018 ECF4A 0000          CON(5) =ZBRNH         IF          ( * [NUM] * )
          0
 1019 ECF4F 3C00          REL(5) %IF89
          0
 1020 ECF54 0000          CON(5) =DUP           DUP         ( * NUM NUM * )
          0
 1021 ECF59 0000          CON(5) =LIT           9
          0
 1022 ECF5E 9000          CON(5) 9
          0
 1023 ECF63 0000          CON(5) =EQUAL         =
          0
 1024 ECF68 0000          CON(5) =ZBRNH         IF          ( * NUM * )
          0
 1025 ECF6D 2300          REL(5) %IF90
          0
 1026             *
 1027             * FIELD SELECT A
```



```
 1028             *   OP2 = 4
 1029             *
 1030 ECF72 0000          CON(5) =DROP          DROP
          0
 1031 ECF77 0000          CON(5) =THREE         3
          0
 1032 ECF7C 0000          CON(5) =aOPLEN        OPLEN
          0
 1033 ECF81 0000          CON(5) =STORE         !           ( * * )
          0
 1034 ECF86 0000          CON(5) =LIT           4
          0
 1035 ECF8B 4000          CON(5) 4
          0
 1036 ECF90 F06C          CON(5) =aOP2          OP2
          E
 1037 ECF95 0000          CON(5) =STORE         !
          0
 1038 ECF9A 0000          CON(5) =SEMI          ;           ( * * )
          0
 1039 ECF9F        %IF90                        ELSE        ( * NUM * )
 1040 ECF9F 0000          CON(5) =DUP           DUP
          0
 1041 ECFA4 0000          CON(5) =LIT           7
          0
 1042 ECFA9 7000          CON(5) 7
          0
 1043 ECFAE 0000          CON(5) =EQUAL         =
          0
 1044 ECFB3 0000          CON(5) =ZBRNH         IF          ( * NUM * )
          0
 1045 ECFB8 6400          REL(5) %IF91
          0
 1046             *
 1047             * FIELD SELECT B
 1048             *   OP2 = 4
 1049             *   OP3 = OP3 + 8
 1050             *
 1051 ECFBD 0000          CON(5) =DROP          DROP
          0
 1052 ECFC2 0000          CON(5) =THREE         3
          0
 1053 ECFC7 0000          CON(5) =aOPLEN        OPLEN
          0
 1054 ECFCC 0000          CON(5) =STORE         !           ( * * )
          0
 1055 ECFD1 0000          CON(5) =LIT           4
          0
 1056 ECFD6 4000          CON(5) 4
          0
 1057 ECFDB F06C          CON(5) =aOP2          OP2
          E
 1058 ECFE0 0000          CON(5) =STORE         !           ( * * )
          0
 1059 ECFE5 0000          CON(5) =LIT           8
```

```
          0
1060 ECFEA 8000          CON(5) 8
          0
1061 ECFEF 326C          CON(5) =aOP3           OP3
          E
1062 ECFF4 0000          CON(5) =PSTOR          +!          ( * * )
          0
1063 ECFF9 0000          CON(5) =SEMI           ;           ( * * )
          0
1064               *
1065               * ALL OTHER FIELD SELECTS
1066               *   OP4 = <FIELD SELECT> - 1
1067               *
1068 ECFFE          %IF91                       ELSE        ( * NUM * )
1069 ECFFE 0000          CON(5) =ONE-           1-
          0
1070 ED003 146C          CON(5) =aOP4           OP4
          E
1071 ED008 0000          CON(5) =STORE          !
          0
1072 ED00D 0000          CON(5) =SEMI           ;           ( * * )
          0
1073               *
1074               * FIELD SELECT FAILED.  LOOK FOR EXPRESSION
1075               *
1076 ED012          %IF89                       ELSE        ( * * )
1077 ED012 919C          CON(5) =aEXP16         1<=EXP<=16
          E
1078 ED017 0000          CON(5) =ZBRNH          IF          ( * NUM * )
          0
1079 ED01C D200          REL(5) %IF92
          0
1080               *
1081               * EXPRESSION IN RANGE [1,16]
1082               *   OP3 = OP3 + 8
1083               *   OP4 = <EXPRESSION> - 1
1084               *
1085 ED021 0000          CON(5) =LIT            8
          0
1086 ED026 8000          CON(5) 8
          0
1087 ED02B 326C          CON(5) =aOP3           OP3
          E
1088 ED030 0000          CON(5) =PSTOR          +!          ( * NUM * )
          0
1089 ED035 0000          CON(5) =ONE-           1-
          0
1090 ED03A 146C          CON(5) =aOP4           OP4
          E
1091 ED03F 0000          CON(5) =STORE          !
          0
1092 ED044 0000          CON(5) =SEMI           ;           ( * * )
          0
1093               *
1094               * NEITHER FIELD SELECT NOR EXPRESSION IN RANGE
```



```
1095               *
1096 ED049          %IF92                       ELSE        ( * NUM * )
1097 ED049 0000          CON(5) =LIT            33
          0
1098 ED04E 1200          CON(5) 33              'illegal transfer value'
          0
1099 ED053 0000          CON(5) =aERROR         ERROR
          0
1100 ED058 0000          CON(5) =DROP           DROP
          0
1101 ED05D 0000          CON(5) =SEMI           ;           ( * * )
          0
1102               ****************************************************************
1103               ***
1104               ***  MAXMIN
1105               ***
1106               *** ( STR NUM -- STR NUM )
1107               *** ( ENSURE LENGTH OF STRING WITHIN LIMITS )
1108               ****************************************************************
1109 ED062 0000 =aMAXMI CON(5) =DOCOL          : MAXMIN
          0
1110 ED067 0000          CON(5) =ZERO           0
          0
1111 ED06C 0000          CON(5) =MAX            MAX
          0
1112 ED071 0000          CON(5) =LIT            16
          0
1113 ED076 0100          CON(5) 16
          0
1114 ED07B 0000          CON(5) =MIN            MIN         ( * STR NUM * )
          0
1115               *
1116               * CONSTANT CAN BE IN [0,16], WHOLE INSTRUCTION INCLUDES OPCODES T
1117               *
1118 ED080 0000          CON(5) =aOPLEN         OPLEN
          0
1119 ED085 0000          CON(5) =AT             @
          0
1120 ED08A 0000          CON(5) =ADD            +
          0
1121 ED08F 0000          CON(5) =SEMI           ;           ( * STR NUM * )
          0
1122               ****************************************************************
1123               ***
1124               ***  CALCLENHEX
1125               ***
1126               *** ( -- STR NUM )
1127               *** ( CALCULATE THE LENGTH OF THE HEX EXPRESSION )
1128               ****************************************************************
1129 ED094 0000 =aCLCHX CON(5) =DOCOL          : CALCLENHEX
          0
1130 ED099 0000          CON(5) =aEXP.F         EXPRESSION.FIELD
          0
1131               *
1132               * DROP LEADING # SIGN (OPTIONAL)
```

```
1133                *
1134 ED09E 0000        CON(5) =LIT          35
          0
1135 ED0A3 3200        CON(5) \#\
          0
1136 ED0A8 0000        CON(5) =aDROPc       DROPc       ( * STR * )
          0
1137 ED0AD 0000        CON(5) =DUP          DUP         ( * STR LEN * )
          0
1138                *
1139                * TRIM STRING TO CORRECT LENGTH
1140                *
1141 ED0B2 260D        CON(5) =aMAXMI       MAXMIN
          E
1142 ED0B7 0000        CON(5) =SEMI         ;           ( * STR LEN * )
          0
1143                ****************************************************************
1144                ***
1145                ***  STORENIB
1146                ***
1147                *** ( POS CHR -- )
1148                *** ( STORE A HEXIDECIMAL CHARACTER AT OPx )
1149                ****************************************************************
1150 ED0BC 0000 =aSTNIB CON(5) =DOCOL        : STORENIB
          0
1151 ED0C1 0000        CON(5) =DUP          DUP
          0
1152                *
1153                * LOOK AT DIGIT
1154                *
1155 ED0C6 0000        CON(5) =aDIG16       ?DIGIT.16
          0
1156 ED0CB 0000        CON(5) =ZBRNH        IF          ( * POS C * )
          0
1157 ED0D0 3200        REL(5) %IF94
          0
1158                *
1159                * ENSURE A HEX DIGIT
1160                *
1161 ED0D5 0000        CON(5) =LIT          16
          0
1162 ED0DA 0100        CON(5) 16
          0
1163 ED0DF 0000        CON(5) =DIGIT        DIGIT
          0
1164 ED0E4 0000        CON(5) =DROP         DROP        ( * POS NUM * )
          0
1165 ED0E9 0000        CON(5) =BRNCH        ELSE
          0
1166 ED0EE E100        REL(5) %TH94
          0
1167                *
1168                * NON - HEX DIGIT PRESENT
1169                *
1170 ED0F3       %IF94                      ELSE        ( * POS C * )
```



```
1171 ED0F3 0000        CON(5) =DROP         DROP
          0
1172 ED0F8 0000        CON(5) =ZERO         0
          0
1173 ED0FD 0000        CON(5) =LIT          34
          0
1174 ED102 2200        CON(5) 34            'non-hexidecimal digit present'
          0
1175 ED107 0000        CON(5) =aERROR       ERROR
          0
1176                *
1177                * VALID DIGIT.  STORE INTO OP[PTR]
1178                *
1179 ED10C       %TH94                      THEN        ( * POS C * )
1180 ED10C 0000        CON(5) =SWAP         SWAP
          0
1181 ED111 0000        CON(5) =a>OPx        >OPx
          0
1182 ED116 0000        CON(5) =STORE        !
          0
1183 ED11B 0000        CON(5) =SEMI         ;           ( * * )
          0
1184                ****************************************************************
1185                ***
1186                ***  LASTCHECKHEX
1187                ***
1188                *** ( STR -- )
1189                *** ( ENSURE WE HAVE NO EXTRA HEX CHARACTERS )
1190                ****************************************************************
1191 ED120 0000 =aLSCHX CON(5) =DOCOL        : LASTCHECKHEX
          0
1192                *
1193                * STRING HAVE LENGTH ZERO (0) ?
1194                *
1195 ED125 0000        CON(5) =ZBRNH        IF
          0
1196 ED12A 4100        REL(5) %IF95
          0
1197                *
1198                * NO.  CHARACTERS STILL REMAIN
1199                *
1200 ED12F 0000        CON(5) =LIT          35
          0
1201 ED134 3200        CON(5) 35            'too many hex digits present'
          0
1202 ED139 0000        CON(5) =aERROR       ERROR
          0
1203                *
1204                * YES.  NO PROBLEM.
1205                *
1206 ED13E       %IF95                      THEN
1207 ED13E 0000        CON(5) =DROP         DROP
          0
1208 ED143 0000        CON(5) =SEMI         ;           ( * * )
          0
```

```
1209           ***************************************************************
1210           ***
1211           ***  NIBHEX
1212           ***
1213           *** ( -- [NUM] )
1214           *** ( NIBHEX STATEMENT )
1215           ***************************************************************
1216 ED148 0000 =aNIBHX CON(5) =DOCOL         : NIBHEX
         0
1217           *
1218           * CALCULATE LENGTH
1219           *
1220 ED14D 490D         CON(5) =aCLCHX        CALCLENHEX  ( * STR NUM * )
         E
1221 ED152 0F9C         CON(5) =a?PAS1        ?PASS=1
         E
1222 ED157 0000         CON(5) =ZBRNH         IF          ( * STR NUM * )
         0
1223 ED15C 9100         REL(5) %IF96
         0
1224           *
1225           * PASS 1 PROCESSING
1226           *
1227 ED161 0000         CON(5) =>R            >R
         0
1228 ED166 0000         CON(5) =2DROP         2DROP
         0
1229 ED16B 0000         CON(5) =R>            R>          ( * NUM * )
         0
1230 ED170 0000         CON(5) =SEMI          ;           ( * NUM * )
         0
1231           *
1232           * PASS 2 PROCESSING
1233           *
1234 ED175      %IF96
1235           *
1236           * SAVE OPCODE LENGTH
1237           *
1238 ED175 0000         CON(5) =DUP           DUP
         0
1239 ED17A 0000         CON(5) =aOPLEN        OPLEN
         0
1240 ED17F 0000         CON(5) =STORE         !           ( * STR NUM * )
         0
1241           *
1242           * LOOP THROUGH EACH CHARACTER
1243           *
1244 ED184 0000         CON(5) =ONE           1
         0
1245 ED189 0000         CON(5) =MAX           MAX
         0
1246 ED18E 0000         CON(5) =ZERO          0
         0
1247 ED193 0000         CON(5) =XDO           DO          ( * STR * )
         0
```



```
1248 ED198      %DO8
1249           *
1250           * EXTRACT CHARACTER & STORE IN APPROPRIATE OPCODE
1251           *
1252 ED198 0000         CON(5) =C@+           C@+
         0
1253 ED19D 0000         CON(5) =R@            I
         0
1254 ED1A2 0000         CON(5) =ONE+          1+
         0
1255 ED1A7 0000         CON(5) =SWAP          SWAP        ( * STR I C * )
         0
1256 ED1AC CB0D         CON(5) =aSTNIB        STORENIB    ( * STR * )
         E
1257 ED1B1 0000         CON(5) =XLOOP         LOOP        ( * STR * )
         0
1258 ED1B6 2EFF         REL(5) %DO8
         F
1259           *
1260           * LOOP UNTIL MAXIMUM CHARACTERS CHECKED, THEN ENSURE NONE
1261           * LEFT OVER.
1262           *
1263 ED1BB 021D         CON(5) =aLSCHX        LASTCHCKHEX ( * * )
         E
1264 ED1C0 0000         CON(5) =SEMI          ;           ( * * )
         0
1265           ***************************************************************
1266           ***
1267           ***  LCHEX
1268           ***
1269           *** ( -- [NUM] )
1270           *** ( LCHEX STATEMENT )
1271           ***************************************************************
1272 ED1C5 0000 =aLCHEX CON(5) =DOCOL         : LCHEX
         0
1273           *
1274           * CALCULATE LENGTH
1275           *
1276 ED1CA 490D         CON(5) =aCLCHX        CALCLENHEX  ( * STR LEN * )
         E
1277 ED1CF 0F9C         CON(5) =a?PAS1        ?PASS=1
         E
1278 ED1D4 0000         CON(5) =ZBRNH         IF          ( * STR LEN * )
         0
1279 ED1D9 9100         REL(5) %IF97
         0
1280           *
1281           * PASS 1 PROCESSING.
1282           *
1283 ED1DE 0000         CON(5) =>R            >R
         0
1284 ED1E3 0000         CON(5) =2DROP         2DROP
         0
1285 ED1E8 0000         CON(5) =R>            R>          ( * LEN * )
         0
```

```
1286 ED1ED 0000         CON(5) =SEMI          ;           ( * LEN * )
           0
1287              *
1288              * PASS 2 PROCESSING.
1289              *
1290 ED1F2         %IF97
1291              *
1292              * SAVE OPCODE LENGTH
1293              *
1294 ED1F2 0000         CON(5) =DUP           DUP
           0
1295 ED1F7 0000         CON(5) =aOPLEN        OPLEN
           0
1296 ED1FC 0000         CON(5) =STORE         !           ( * STR LEN * )
           0
1297              *
1298              * SET VARIABLE REVERS BECAUSE DIGITS ARE REVERSED!
1299              *
1300 ED201 0000         CON(5) =DUP           DUP
           0
1301 ED206 0000         CON(5) =aREVER        REVERS
           0
1302 ED20B 0000         CON(5) =STORE         !
           0
1303 ED210 0000         CON(5) =TWO-          2-
           0
1304 ED215 0000         CON(5) =ONE           1
           0
1305 ED21A 0000         CON(5) =MAX           MAX
           0
1306 ED21F 0000         CON(5) =ZERO          0
           0
1307 ED224 0000         CON(5) =XDO           DO          ( * STR * )
           0
1308 ED229         %DO9
1309              *
1310              * LOOP THROUGH EACH CHARACTER
1311              *
1312 ED229 0000         CON(5) =C@+           C@+
           0
1313 ED22E 0000         CON(5) =aREVER        REVERS
           0
1314 ED233 0000         CON(5) =AT            @
           0
1315 ED238 0000         CON(5) =R@            I
           0
1316              *
1317              * CALCULATE APPROPRIATE OPCODE
1318              *
1319 ED23D 0000         CON(5) =MINUS         -           ( * STR C x * )
           0
1320 ED242 0000         CON(5) =SWAP          SWAP
           0
1321              *
1322              * STORE NIBBLE AWAY
```



```
1323              *
1324 ED247 CB0D         CON(5) =aSINIB        STORENIB    ( * STR * )
           E
1325 ED24C 0000         CON(5) =XLOOP         LOOP
           0
1326 ED251 8DFF         REL(5) %DO9
           F
1327              *
1328              * LOOP THROUGH MAXIMUM CHARACTERS ALLOWED, THEN CHECK IF
1329              * ANY LEFT OVER
1330              *
1331 ED256 021D         CON(5) =aLSCHX        LASTCHECKHEX  ( * * )
           E
1332              *
1333              * STORE LENGTH OF EXPRESSION IN OP2
1334              *
1335 ED25B 0000         CON(5) =aREVER        REVERS
           0
1336 ED260 0000         CON(5) =AT            @
           0
1337 ED265 0000         CON(5) =THREE         3
           0
1338 ED26A 0000         CON(5) =MINUS         -
           0
1339 ED26F 0000         CON(5) =ZERO          0
           0
1340 ED274 0000         CON(5) =MAX           MAX
           0
1341 ED279 F06C         CON(5) =aOP2          OP2
           E
1342 ED27E 0000         CON(5) =STORE         !           ( * * )
           0
1343 ED283 0000         CON(5) =SEMI          ;           ( * * )
           0
1344              ****************************************************************
1345              ***
1346              ***  Dx=HEX
1347              ***
1348              *** ( -- )
1349              *** ( D0=HEX AND D1=HEX STATEMENTS )
1350              ****************************************************************
1351 ED288 0000 =Dx=HX  CON(5) =DOCOL         : Dx=HEX
           0
1352 ED28D 0000         CON(5) =aEXP.F        EXPRESSION.FIELD
           0
1353              *
1354              * DROP LEADING # (OPTIONAL)
1355              *
1356 ED292 0000         CON(5) =LIT           35
           0
1357 ED297 3200         CON(5) \#\
           0
1358 ED29C 0000         CON(5) =aDROPc        DROPc       ( * STR * )
           0
1359              *
```

```
1360            * STORE FIRST CHARACTER IN OP4
1361            *
1362 ED2A1 0000         CON(5) =C@+             C@+
         0
1363 ED2A6 0000         CON(5) =LIT             4
         0
1364 ED2AB 4000         CON(5) 4
         0
1365 ED2B0 0000         CON(5) =SWAP            SWAP
         0
1366 ED2B5 CB0D         CON(5) =aSINIB          STORENIB    ( * STR * )
         E
1367            *
1368            * STORE SECOND CHARACTER IN OP3
1369            *
1370 ED2BA 0000         CON(5) =C@+             C@+
         0
1371 ED2BF 0000         CON(5) =THREE           3
         0
1372 ED2C4 0000         CON(5) =SWAP            SWAP
         0
1373 ED2C9 CB0D         CON(5) =aSINIB          STORENIB    ( * STR * )
         E
1374            *
1375            * ENSURE NO EXTRA CHARACTERS
1376            *
1377 ED2CE 021D         CON(5) =aLSCHX          LASTCHECKHEX
         E
1378 ED2D3 0000         CON(5) =SEMI            ;           ( * * )
         0
1379            ****************************************************************
1380            ***
1381            ***  LASTCHR
1382            ***
1383            *** ( STR -- STR C )
1384            *** ( GET A COPY OF THE LAST CHARACTER OF THE STRING )
1385            ****************************************************************
1386 ED2D8 0000 =aLSTCH CON(5) =DOCOL         : LASTCHR
         0
1387 ED2DD 0000         CON(5) =DUP             DUP         ( * STR LEN * )
         0
1388 ED2E2 0000         CON(5) =ZBRNH           IF          ( * STR * )
         0
1389 ED2E7 3200         REL(5) %IF104
         0
1390            *
1391            * GO TO END OF STRING AND GET THAT CHARACTER
1392            *
1393 ED2EC 0000         CON(5) =2DUP            2DUP
         0
1394 ED2F1 0000         CON(5) =ONE-            1-
         0
1395 ED2F6 0000         CON(5) =TWO*            2*
         0
1396 ED2FB 0000         CON(5) =ADD             +
```



```
         0
1397 ED300 0000         CON(5) =CAT             C@
         0
1398 ED305 0000         CON(5) =SEMI            ;           ( * STR C * )
         0
1399            *
1400            * NULL LENGTH STRING
1401            *
1402 ED30A       %IF104                         ELSE        ( * STR * )
1403 ED30A 0000         CON(5) =ZERO            0
         0
1404 ED30F 0000         CON(5) =SEMI            ;           ( * STR C * )
         0
1405            ****************************************************************
1406            ***
1407            ***  QUOTED.STRING
1408            ***
1409            *** ( STR -- STR )
1410            *** ( REMOVE SINGLE QUOTE MARKS FROM FRONT AND BACK )
1411            ****************************************************************
1412 ED314 0000 =aQ"STR CON(5) =DOCOL         : QUOTED.STRING
         0
1413            *
1414            * LOOK AT FIRST CHARACTER
1415            *
1416 ED319 0000         CON(5) =OVER            OVER
         0
1417 ED31E 0000         CON(5) =CAT             C@
         0
1418 ED323 0000         CON(5) =LIT             39
         0
1419 ED328 7200         CON(5) \'\
         0
1420 ED32D 0000         CON(5) =EQUAL           =
         0
1421 ED332 0000         CON(5) =OVER            OVER
         0
1422 ED337 0000         CON(5) =AND             AND         ( * STR F * )
         0
1423 ED33C 0000         CON(5) =DUP             DUP
         0
1424            *
1425            * SAVE FLAG ON RETURN STACK
1426            *
1427 ED341 0000         CON(5) =>R              >R
         0
1428 ED346 0000         CON(5) =ZBRNH           IF          ( * STR * )
         0
1429 ED34B F000         REL(5) %IF99
         0
1430            *
1431            * FIRST CHARACTER IS A '
1432            *
1433 ED350 0000         CON(5) =C@+             C@+
         0
```

```
1434 ED355 0000         CON(5) =DROP          DROP        ( * STR * )
           0
1435               *
1436               * LOOK AT LAST CHARACTER
1437               *
1438 ED35A          %IF99                     THEN        ( * STR * )
1439 ED35A 8D2D     E        CON(5) =aLSTCH       LASTCHR
1440 ED35F 0000             CON(5) =LIT          39
           0
1441 ED364 7200             CON(5) \'\
           0
1442 ED369 0000             CON(5) =EQUAL        =           ( * STR F * )
           0
1443 ED36E 0000             CON(5) =DUP          DUP
           0
1444               *
1445               * SAVE FLAG ON RETURN STACK
1446               *
1447 ED373 0000             CON(5) =>R           >R
           0
1448 ED378 0000             CON(5) =ZBRNH        IF          ( * STR * )
           0
1449 ED37D A000             REL(5) %IF100
           0
1450               *
1451               * LAST CHARACTER IS A '
1452               *
1453 ED382 0000             CON(5) =ONE-         1-
           0
1454               *
1455               * LOOK AT FLAGS.  IF EITHER FALSE, THEN REPORT ERROR
1456               *
1457 ED387          %IF100                    THEN        ( * STR * )
1458 ED387 0000             CON(5) =R>           R>
           0
1459 ED38C 0000             CON(5) =R>           R>          ( * STR F F' * )
           0
1460 ED391 0000             CON(5) =AND          AND
           0
1461 ED396 0000             CON(5) =ZEQ          0=
           0
1462 ED39B 0000             CON(5) =ZBRNH        IF          ( * STR * )
           0
1463 ED3A0 4100             REL(5) %IF98
           0
1464 ED3A5 0000             CON(5) =LIT          24
           0
1465 ED3AA 8100             CON(5) 24            'invalid quoted string'
           0
1466 ED3AF 0000             CON(5) =aP2ERR       PASS 2 ERROR
           0
1467               *
1468               * ALL DONE]
1469               *
```



```
1470 ED3B4          %IF98                     THEN        ( * STR * )
1471 ED3B4 0000             CON(5) =SEMI         ;           ( * STR * )
           0
1472               ********************************************************************
1473               ***
1474               ***  TRIM.BLANKS
1475               ***
1476               *** ( STR -- STR )
1477               *** ( TRIM LEADING & TRAILING SPACES.  IGNORE CHARS AFTER )
1478               *** ( ANY INTERNAL SPACE )
1479               ********************************************************************
1480 ED3B9 0000 =aTRIM  CON(5) =DOCOL        : TRIM.BLANKS
           0
1481               *
1482               * TACK A TRAILING ZERO ON STRING, SO ENCLOSE SUCCEEDS IN THAT CAS
1483               *
1484 ED3BE 0000             CON(5) =TWO*         2*          ( * ADDR NIBS * )
           0
1485 ED3C3 0000             CON(5) =OVER         OVER        ( * ADDR NIBS ADDR * )
           0
1486 ED3C8 0000             CON(5) =ADD          +           ( * ADDR ADDR' * )
           0
1487 ED3CD 0000             CON(5) =ZERO         0
           0
1488 ED3D2 0000             CON(5) =SWAP         SWAP        ( * ADDR 0 ADDR' * )
           0
1489 ED3D7 0000             CON(5) =C!           C!          ( * ADDR * )
           0
1490 ED3DC 0000             CON(5) =LIT          32
           0
1491 ED3E1 0200             CON(5) 32
           0
1492               *
1493               * ENCLOSE STRING USING BLANKS AS DELIMITER
1494               *
1495 ED3E6 0000             CON(5) =ENCL         ENCLOSE     ( * ADDR N1 N2 N3 * )
           0
1496               *
1497               * CALCULATE RESULTING STRING
1498               *   ADDR = ADDR + N1
1499               *   LEN = (N1 - MIN(N2,N3))/2
1500               *
1501 ED3EB 0000             CON(5) =MIN          MIN         ( * ADDR N1 N2 * )
           0
1502 ED3F0 0000             CON(5) =OVER         OVER        ( * ADDR N1 N2 N1 * )
           0
1503 ED3F5 0000             CON(5) =MINUS        -
           0
1504 ED3FA 0000             CON(5) =TWO/         2/          ( * ADDR N1 LEN * )
           0
1505 ED3FF 0000             CON(5) =>R           >R          ( * ADDR N1 * )
           0
1506 ED404 0000             CON(5) =ADD          +           ( * ADDR * )
           0
1507 ED409 0000             CON(5) =R>           R>          ( * STR * )
```

```
         0
1508 ED40E 0000         CON(5) =SEMI            ;           ( * STR * )
         0
1509            ****************************************************************
1510            ***
1511            ***  CALCLENASC
1512            ***
1513            *** ( -- STR LEN )
1514            *** ( CALCULATE THE LENGTH OF THE ASCII EXPRESSION )
1515            ****************************************************************
1516 ED413 0000 =aCLCFF CON(5) =DOCOL          : CALCLENASC
         0
1517 ED418 0000         CON(5) =aEXP.F          EXPRESSION  ( * STR * )
         0
1518 ED41D 413D         CON(5) =aQ"STR          QUOTED      ( * STR * )
         E
1519 ED422 0000         CON(5) =DUP             DUP
         0
1520 ED427 0000         CON(5) =TWO*            2*
         0
1521 ED42C 260D         CON(5) =aMAXMI          MAXMIN
         E
1522 ED431 0000         CON(5) =SEMI            ;           ( * STR LEN * )
         0
1523            ****************************************************************
1524            ***
1525            ***  STARTLOOP
1526            ***
1527            *** ( STR LEN -- STR HIGH LOW )
1528            *** ( START THE CHARACTER BY CHARACTER LOOP )
1529            ****************************************************************
1530 ED436 0000 =aSTLUP CON(5) =DOCOL          : STARTLOOP
         0
1531 ED43B 0000         CON(5) =DUP             DUP
         0
1532 ED440 0000         CON(5) =aOPLEN          OPLEN
         0
1533 ED445 0000         CON(5) =STORE           !           ( * STR LEN * )
         0
1534 ED44A 0000         CON(5) =aREVER          REVERS
         0
1535 ED44F 0000         CON(5) =STORE           !
         0
1536            *
1537            * ENSURE NULL STRING NOT PRESENT
1538            *
1539 ED454 0000         CON(5) =DUP             DUP
         0
1540 ED459 0000         CON(5) =ZEQ             0=
         0
1541 ED45E 0000         CON(5) =ZBRNH           IF
         0
1542 ED463 4100         REL(5) %IF58
         0
1543 ED468 0000         CON(5) =LIT             39
```



```
         0
1544 ED46D 7200         CON(5) 39               'illegal expression'
         0
1545 ED472 0000         CON(5) =aERROR          ERROR
         0
1546 ED477       %IF58                          THEN        ( * STR * )
1547            *
1548            * SET UP LOOP PARAMETERS
1549            *
1550 ED477 0000         CON(5) =DUP             DUP         ( * STR LEN * )
         0
1551 ED47C 0000         CON(5) =LIT             8
         0
1552 ED481 8000         CON(5) 8
         0
1553 ED486 0000         CON(5) =MIN             MIN
         0
1554 ED48B 0000         CON(5) =ONE             1
         0
1555 ED490 0000         CON(5) =MAX             MAX
         0
1556 ED495 0000         CON(5) =ZERO            0
         0
1557 ED49A 0000         CON(5) =SEMI            ;           ( * STR HIGH LOW * )
         0
1558            ****************************************************************
1559            ***
1560            ***  LASTCHECKASC
1561            ***
1562            *** ( STR -- )
1563            *** ( ENSURE NO TRAILING CHARACTERS IN QUOTED STRING )
1564            ****************************************************************
1565 ED49F 0000 =aLSCFF CON(5) =DOCOL          : LASTCHECKASC
         0
1566            *
1567            * NULL STRING ?
1568            *
1569 ED4A4 0000         CON(5) =ZBRNH           IF
         0
1570 ED4A9 4100         REL(5) %IF101
         0
1571            *
1572            * NO.  REPORT ERROR
1573            *
1574 ED4AE 0000         CON(5) =LIT             36
         0
1575 ED4B3 4200         CON(5) 36               'too many ascii chars present'
         0
1576 ED4B8 0000         CON(5) =aERROR          ERROR
         0
1577            *
1578            * YES.  ALL OK
1579            *
1580 ED4BD       %IF101                         THEN
1581 ED4BD 0000         CON(5) =DROP            DROP
```

```
         0
1582 ED4C2 0000          CON(5) =SEMI          ;          ( * * )
         0
1583             ***************************************************************
1584             ***
1585             ***  NIBASC
1586             ***
1587             *** ( -- [NUM] )
1588             *** ( NIBASC STATEMENT )
1589             ***************************************************************
1590 ED4C7 0000 -aNIBFF CON(5) -DOCOL          : NIBASC
         0
1591             *
1592             * CALCULATE LENGTH OF INSTRUCTION
1593             *
1594 ED4CC 314D          CON(5) =aCLCFF        CALCLENASC  ( * STR LEN * )
         E
1595 ED4D1 0F9C          CON(5) =a?PAS1        ?PASS=1
         E
1596 ED4D6 0000          CON(5) =ZBRNH         IF          ( * STR LEN * )
         0
1597 ED4DB 9100          REL(5) %IF102
         0
1598             *
1599             * PASS 1 PROCESSING
1600             *
1601 ED4E0 0000          CON(5) =>R            >R
         0
1602 ED4E5 0000          CON(5) =2DROP         2DROP
         0
1603 ED4EA 0000          CON(5) =R>            R>          ( * NUM * )
         0
1604 ED4EF 0000          CON(5) =SEMI          ;           ( * NUM * )
         0
1605             *
1606             * PASS 2 PROCESSING
1607             *
1608 ED4F4        %IF102                       ELSE        ( * NUM * )
1609 ED4F4 634D          CON(5) =aSTLUP        STARTLOOP
         E
1610 ED4F9 0000          CON(5) =XDO           DO          ( * STR * )
         0
1611             *
1612             * LOOP THROUGH EACH CHARACTER IN EXPRESSION FIELD
1613             *
1614 ED4FE        %DO10
1615 ED4FE 0000          CON(5) =C@+           C@+
         0
1616             *
1617             * CALCULATE LEAST SIGNIFICANT NIBBLE
1618             *
1619 ED503 0000          CON(5) =DUP           DUP         ( * STR C C * )
         0
1620 ED508 0000          CON(5) =LIT           16
         0
```



```
1621 ED50D 0100          CON(5) 16
         0
1622 ED512 0000          CON(5) =/             /
         0
1623 ED517 0000          CON(5) =DUP           DUP         ( * STR C N N * )
         0
1624             *
1625             * CALCULATE OPCODE NIBBLE ADDRESS
1626             *
1627 ED51C 0000          CON(5) =R@            I
         0
1628 ED521 0000          CON(5) =ONE+          1+
         0
1629 ED526 0000          CON(5) =TWO*          2*          ( * STR C N N PTR * )
         0
1630 ED52B 0000          CON(5) =a>OPx         >OPx
         0
1631             *
1632             * SAVE POSITION ON RETURN STACK
1633             *
1634 ED530 0000          CON(5) =DUP           DUP
         0
1635 ED535 0000          CON(5) =>R            >R
         0
1636             *
1637             * STORE LEAST SIGNIFICANT NIBBLE
1638             *
1639 ED53A 0000          CON(5) =STORE         !           ( * STR C N * )
         0
1640 ED53F 0000          CON(5) =LIT           16
         0
1641 ED544 0100          CON(5) 16
         0
1642 ED549 0000          CON(5) =MULT          *
         0
1643 ED54E 0000          CON(5) =MINUS         -           ( * STR N * )
         0
1644             *
1645             * STORE MOST SIGNIFICANT NIBBLE
1646             *
1647 ED553 0000          CON(5) =R>            R>
         0
1648 ED558 0000          CON(5) =FIVE-         5-
         0
1649 ED55D 0000          CON(5) =STORE         !           ( * STR * )
         0
1650 ED562 0000          CON(5) =XLOOP         LOOP        ( * STR * )
         0
1651 ED567 79FF          REL(5) %DO10
         F
1652 ED56C F94D          CON(5) =aLSCFF        LASTCHECKASC
         E
1653 ED571 0000          CON(5) =SEMI          ;           ( * * )
         0
1654             ***************************************************************
```

```
1655            ***
1656            ***  LCASC
1657            ***
1658            *** ( -- [NUM] )
1659            *** ( LCASC STATEMENT )
1660            ********************************************************************
1661 ED576 0000 =aLCASC CON(5) =DOCOL          : LCASC
         0
1662            *
1663            * CALCULATE LENGTH OF INSTRUCTION
1664            *
1665 ED57B 314D         CON(5) =aCLCFF         CALCLENASC  ( * STR LEN * )
         E
1666 ED580 0F9C         CON(5) =a?PAS1         ?PASS=1
         E
1667 ED585 0000         CON(5) =ZBRNH          IF          ( * STR LEN * )
         0
1668 ED58A 9100         REL(5) %IF103
         0
1669            *
1670            * PASS 1 PROCESSING
1671            *
1672 ED58F 0000         CON(5) =>R             >R
         0
1673 ED594 0000         CON(5) =2DROP          2DROP
         0
1674 ED599 0000         CON(5) =R>             R>          ( * NUM * )
         0
1675 ED59E 0000         CON(5) =SEMI           ;           ( * NUM * )
         0
1676            *
1677            * PASS 2 PROCESSING
1678            *
1679 ED5A3       %IF103                                    ( * NUM * )
1680 ED5A3 634D         CON(5) =aSTLUP         STARTLOOP
         E
1681 ED5A8 0000         CON(5) =XDO            DO          ( * STR * )
         0
1682            *
1683            * LOOP THROUGH EACH CHARACTER
1684            *
1685 ED5AD       %DO11
1686 ED5AD 0000         CON(5) =C@+            C@+
         0
1687            *
1688            * CALCULATE LEAST SIGNIFICANT NIBBLE
1689            *
1690 ED5B2 0000         CON(5) =DUP            DUP         ( * STR C C * )
         0
1691 ED5B7 0000         CON(5) =LIT            16
         0
1692 ED5BC 0100         CON(5) 16
         0
1693 ED5C1 0000         CON(5) =/              /
         0
```



```
1694            *
1695            * CALCULATE OPCODE POSITION
1696            *
1697 ED5C6 0000         CON(5) =DUP            DUP
         0
1698 ED5CB 0000         CON(5) =aREVER         REVERS
         0
1699 ED5D0 0000         CON(5) =AT             @           ( * STR C N N MAX * )
         0
1700 ED5D5 0000         CON(5) =R@             I
         0
1701 ED5DA 0000         CON(5) =TWO*           2*
         0
1702 ED5DF 0000         CON(5) =MINUS          -
         0
1703 ED5E4 0000         CON(5) =LIT            4
         0
1704 ED5E9 4000         CON(5) 4
         0
1705 ED5EE 0000         CON(5) =MAX            MAX
         0
1706 ED5F3 0000         CON(5) =a>OPx          >OPx        ( * STR C N N PTR * )
         0
1707            *
1708            * SAVE OPCODE POSITION ON RETURN STACK
1709            *
1710 ED5F8 0000         CON(5) =DUP            DUP
         0
1711 ED5FD 0000         CON(5) =>R             >R
         0
1712            *
1713            * STORE LEAST SIGNIFICANT NIBBLE
1714            *
1715 ED602 0000         CON(5) =STORE          !           ( * STR C N * )
         0
1716            *
1717            * CALCULATE MOST SIGNIFICANT NIBBLE
1718            *
1719 ED607 0000         CON(5) =LIT            16
         0
1720 ED60C 0100         CON(5) 16
         0
1721 ED611 0000         CON(5) =MULT           *
         0
1722 ED616 0000         CON(5) =MINUS          -
         0
1723            *
1724            * STORE MOST SIGNIFICANT NIBBLE
1725            *
1726 ED61B 0000         CON(5) =R>             R>
         0
1727 ED620 0000         CON(5) =FIVE-          5-
         0
1728 ED625 0000         CON(5) =STORE          !           ( * STR * )
         0
```

```
1729 ED62A 0000          CON(5) =XLOOP          LOOP          ( * STR * )
           0
1730 ED62F E7FF          REL(5) %DO11
           F
1731             *
1732             * ENSURE NO EXTRA CHARACTERS
1733             *
1734 ED634 F94D          CON(5) =aLSCFF         LASTCHEKASC ( * * )
           E
1735             *
1736             * STORE CONSTANT LENGTH IN OP2
1737             *
1738 ED639 0000          CON(5) =aREVER         REVERS
           0
1739 ED63E 0000          CON(5) =AT             @
           0
1740 ED643 0000          CON(5) =THREE          3
           0
1741 ED648 0000          CON(5) =MINUS          -
           0
1742 ED64D 0000          CON(5) =ZERO           0
           0
1743 ED652 0000          CON(5) =MAX            MAX
           0
1744 ED657 F06C          CON(5) =aOP2           OP2
           E
1745 ED65C 0000          CON(5) =STORE          !             ( * * )
           0
1746 ED661 0000          CON(5) =SEMI           ;             ( * * )
           0
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER


```
1747                     STITLE FIT STATS
1748             zSIZE   EQU    (*)-zTHIS
1749             zLEFT   EQU    (zNEXT)-*
1750 ED666               BSS    zLEFT
1751 ED700               END
```

$RANGE	Abs	969040	#EC950 -	379	378
%D010	Abs	972030	#ED4FE -	1614	1651
%D011	Abs	972205	#ED5AD -	1685	1730
%D07	Abs	969135	#EC9AF -	441	458
%D08	Abs	971160	#ED198 -	1248	1258
%D09	Abs	971305	#ED229 -	1308	1326
%IF100	Abs	971655	#ED387 -	1457	1449
%IF101	Abs	971965	#ED4BD -	1580	1570
%IF102	Abs	972020	#ED4F4 -	1608	1597
%IF103	Abs	972195	#ED5A3 -	1679	1668
%IF104	Abs	971530	#ED30A -	1402	1389
%IF58	Abs	971895	#ED477 -	1546	1542
%IF59	Abs	968488	#EC728 -	155	147
%IF60	Abs	968539	#EC75B -	171	163
%IF61	Abs	968636	#EC7BC -	202	194
%IF62	Abs	968687	#EC7EF -	218	210
%IF63	Abs	968738	#EC822 -	234	226
%IF64	Abs	968789	#EC855 -	250	242
%IF65	Abs	968585	#EC789 -	186	179
%IF66	Abs	968837	#EC885 -	265	258
%IF67	Abs	968885	#EC8B5 -	280	273
%IF68	Abs	969405	#ECABD -	542	494
%IF69	Abs	969315	#ECA63 -	516	503
%IF70	Abs	969400	#ECAB8 -	537	532
%IF71	Abs	969495	#ECB17 -	581	557
%IF72	Abs	969485	#ECB0D -	575	566
%IF73	Abs	969740	#ECC0C -	650	588
%IF74	Abs	969625	#ECB99 -	623	594
%IF75	Abs	969850	#ECC7A -	695	669
%IF76	Abs	969830	#ECC66 -	687	675
%IF77	Abs	969925	#ECCC5 -	728	718
%IF78	Abs	970040	#ECD38 -	776	736
%IF79	Abs	969995	#ECD0B -	756	747
%IF80	Abs	970070	#ECD56 -	786	772
%IF81	Abs	970140	#ECD9C -	818	808
%IF82	Abs	970180	#ECDC4 -	843	833
%IF83	Abs	970240	#ECE00 -	872	862
%IF84	Abs	970310	#ECE46 -	904	891
%IF85	Abs	970370	#ECE82 -	933	923
%IF86	Abs	970435	#ECEC3 -	963	952
%IF87	Abs	970565	#ECF45 -	1013	983
%IF88	Abs	970550	#ECF36 -	1006	997
%IF89	Abs	970770	#ED012 -	1076	1019
%IF90	Abs	970655	#ECF9F -	1039	1025
%IF91	Abs	970750	#ECFFE -	1068	1045
%IF92	Abs	970825	#ED049 -	1096	1079
%IF94	Abs	970995	#ED0F3 -	1170	1157
%IF95	Abs	971070	#ED13E -	1206	1196
%IF96	Abs	971125	#ED175 -	1234	1223
%IF97	Abs	971250	#ED1F2 -	1290	1279
%IF98	Abs	971700	#ED3B4 -	1470	1463
%IF99	Abs	971610	#ED35A -	1438	1429
%TH59	Abs	968890	#EC8BA -	290	151
%TH60	Abs	968890	#EC8BA -	289	167
%TH61	Abs	968890	#EC8BA -	288	198

%TH62	Abs	968890	#EC8BA -	287	214						
%TH63	Abs	968890	#EC8BA -	286	230						
%TH64	Abs	968890	#EC8BA -	285	246						
%TH65	Abs	968890	#EC8BA -	284	182						
%TH66	Abs	968890	#EC8BA -	283	261						
%TH67	Abs	968890	#EC8BA -	282	276						
%TH79	Abs	970010	#ECD1A -	763	751						
%TH94	Abs	971020	#ED10C -	1179	1166						
/	Ext		-	1622	1693						
2DROP	Ext		-	295	1228	1284	1602	1673			
2DUP	Ext		-	140	156	172	187	203	219	235	251
				266	1393						
>R	Ext		-	294	1227	1283	1427	1447	1505	1601	1635
				1672	1711						
?DUP	Ext		-	492	586	667	1017				
ADD	Ext		-	42	53	435	509	612	634	682	764
				1120	1396	1486	1506				
AND	Ext		-	65	337	366	716	1422	1460		
AT	Ext		-	64	473	508	528	605	609	628	713
				741	749	758	769	778	780	805	897
				1119	1314	1336	1699	1739			
BRNCH	Ext		-	150	166	181	197	213	229	245	260
				275	750	1165					
C!	Ext		-	1489							
C@+	Ext		-	1252	1312	1362	1370	1433	1615	1686	
CAT	Ext		-	1397	1417						
DIGIT	Ext		-	1163							
DOCOL	Ext		-	21	29	38	49	62	74	85	96
				108	120	132	305	316	346	433	471
				484	554	662	707	799	827	856	885
				917	946	976	1109	1129	1150	1191	1216
				1272	1351	1386	1412	1480	1516	1530	1565
				1590	1661						
DROP	Ext		-	462	600	787	844	873	905	937	967
				1030	1051	1100	1164	1171	1207	1434	1581
DUP	Ext		-	321	354	445	498	589	629	637	670
				987	1020	1040	1137	1151	1238	1294	1300
				1387	1423	1443	1519	1531	1539	1550	1619
				1623	1634	1690	1697	1710			
=Dx=HX	Abs	971400	#ED288 -	1351							
ENCL	Ext		-	1495							
EQUAL	Ext		-	475	501	530	564	592	640	673	990
				994	1023	1043	1420	1442			
FAIL	Abs	969102	#EC98E -	423	407	418					
FIVE+	Ext		-	31							
FIVE-	Ext		-	1648	1727						
GT	Ext		-	326							
GTZ	Ext		-	355							
LIT	Ext		-	40	51	97	109	121	148	164	195
				211	227	243	331	360	447	499	517
				533	543	562	590	610	632	641	651
				671	680	696	722	788	812	845	874
				906	934	964	988	992	1007	1021	1034
				1041	1055	1059	1085	1097	1112	1134	1161
				1173	1200	1356	1363	1418	1440	1464	1490

				1543	1551	1574	1620	1640	1691	1703	1719	
LT	Ext		-	333	362							
LTZ	Ext		-	446								
MAX	Ext		-	1111	1245	1305	1340	1555	1705	1743		
MIN	Ext		-	1114	1501	1553						
MINUS	Ext		-	1319	1338	1503	1643	1702	1722	1741		
MULT	Ext		-	643	1642	1721						
ONE	Ext		-	75	180	474	1244	1304	1554			
ONE+	Ext		-	1254	1628							
ONE-	Ext		-	521	624	630	688	759	956	1069	1089	
				1394	1453							
OR	Ext		-	995								
OVER	Ext		-	327	356	434	1416	1421	1485	1502		
POS	Abs	969081 #EC979	-	406	401							
PSTOR	Ext		-	536	645	1062	1088					
QUOTC	Ext		-	141	157	173	188	204	220	236	252	
				267								
R>	Ext		-	296	1229	1285	1458	1459	1507	1603	1647	
				1674	1726							
R@	Ext		-	454	1253	1315	1627	1700				
S=	Ext		-	145	161	177	192	208	224	240	256	
				271								
SEMI	Ext		-	23	32	43	54	66	77	88	100	
				112	124	297	308	338	367	463	476	
				512	538	546	571	577	615	646	654	
				691	699	782	791	819	839	848	868	
				877	900	909	929	938	959	968	1002	
				1009	1038	1063	1072	1092	1101	1121	1142	
				1183	1208	1230	1264	1286	1343	1378	1398	
				1404	1471	1508	1522	1557	1582	1604	1653	
				1675	1746							
SHIFT	Abs	969087 #EC97F	-	411	413							
STORE	Ext		-	456	511	520	523	603	607	614	626	
				636	690	838	867	899	928	958	1033	
				1037	1054	1058	1071	1091	1182	1240	1296	
				1302	1342	1533	1535	1639	1649	1715	1728	
				1745								
SUCC	Abs	969100 #EC98C	-	422								
SWAP	Ext		-	436	453	991	1180	1255	1320	1365	1372	
				1488								
THREE	Ext		-	259	576	1001	1031	1052	1337	1371	1740	
TWO	Ext		-	86	274	529	570	601				
TWO*	Ext		-	639	1395	1484	1520	1629	1701			
TWO-	Ext		-	1303								
TWO/	Ext		-	631	638	1504						
UDIV	Ext		-	449								
XDO	Ext		-	437	1247	1307	1610	1681				
XLOOP	Ext		-	457	1257	1325	1650	1729				
ZBRNH	Ext		-	146	162	178	193	209	225	241	257	
				272	493	502	531	556	565	587	593	
				668	674	717	735	746	771	807	832	
				861	890	922	951	982	996	1018	1024	
				1044	1078	1156	1195	1222	1278	1388	1428	
				1448	1462	1541	1569	1596	1667			
ZEQ	Ext		-	714	734	806	1461	1540				

ZERO	Ext		-	281	1110	1172	1246	1306	1339	1403	1487	
				1556	1742							
a>OPx	Ext		-	455	898	1181	1630	1706				
=a?PAS1	Abs	969200 #EC9F0	-	471	555	981	1221	1277	1595	1666		
aARGRP	Ext		-	608	627							
=aBRANC	Abs	969870 #ECC8E	-	707								
=aCLCFF	Abs	971795 #ED413	-	1516	1594	1665						
=aCLCHX	Abs	970900 #ED094	-	1129	1220	1276						
=aDATAT	Abs	970460 #ECEDC	-	976								
aDIG16	Ext		-	1155								
=aDPARI	Abs	970395 #ECE9B	-	946								
aDROPc	Ext		-	1136	1358							
aERROR	Ext		-	545	653	698	724	790	814	847	876	
				908	936	966	1099	1175	1202	1545	1576	
aEXP.F	Ext		-	136	306	1130	1352	1517				
=aEXP15	Abs	968930 #EC8E2	-	316	831	860	889	921				
=aEXP16	Abs	968985 #EC919	-	346	950	1077						
=aEXPR	Abs	968910 #EC8CE	-	305	320	350	729					
aEXPRS	Ext		-	307								
aFILEN	Ext		-	768	779							
aFILMK	Ext		-	757	777							
=aFILOP	Abs	969110 #EC996	-	433	781							
=aFS	Abs	968427 #EC6EB	-	132	488	561	582	663	980			
=aGFLAG	Abs	968287 #EC65F	-	62	76	87	99	111	123			
=aISABS	Abs	968377 #EC6B9	-	108	733							
=aISGSB	Abs	968402 #EC6D2	-	120	745							
=aISGYS	Abs	968352 #EC6A0	-	96	715							
aLC	Ext		-	740								
=aLCASC	Abs	972150 #ED576	-	1661								
=aLCHEX	Abs	971205 #ED1C5	-	1272								
=aLENST	Abs	968312 #EC678	-	74								
=aLSCFF	Abs	971935 #ED49F	-	1565	1652	1734						
=aLSCHX	Abs	971040 #ED120	-	1191	1263	1331	1377					
=aLSTCH	Abs	971480 #ED2D8	-	1386	1439							
aLSTRQ	Ext		-	712	804							
=aMAXMI	Abs	970850 #ED062	-	1109	1141	1521						
aNEG1	Ext		-	325								
=aNIBFF	Abs	971975 #ED4C7	-	1590								
=aNIBHX	Abs	971080 #ED148	-	1216								
aOP	Ext		-	22	30	39	50					
=aOP1	Abs	968192 #EC600	-	21	519	613	635					
=aOP2	Abs	968207 #EC60F	-	29	510	522	535	606	625	644	1036	
				1057	1341	1744						
=aOP3	Abs	968227 #EC623	-	38	604	689	837	866	927	957	1061	
				1087								
=aOP4	Abs	968257 #EC641	-	49	1070	1090						
aOPFLG	Ext		-	63								
aOPLEN	Ext		-	602	748	896	1032	1053	1118	1239	1295	
				1532								
aP2ERR	Ext		-	1466								
aPASS	Ext		-	472								
=aPTRTE	Abs	970145 #ECDA1	-	827								
=aQ"STR	Abs	971540 #ED314	-	1412	1518							
=aRANGE	Abs	969035 #EC94B	-	378	770							
=aREGAR	Abs	969425 #ECAD1	-	554								

```
=aREGLO  Abs  969760 #ECC20 -   662
 aREGRP  Ext                -   507   527
=aREGTE  Abs  969230 #ECA0E -   484
 aREVER  Ext                -  1301  1313  1335  1534  1698  1738
=aRQYES  Abs  968332 #EC68C -    85
=aRTNYS  Abs  970095 #ECD6F -   799
=aSETPT  Abs  970265 #ECE19 -   885
=aSETST  Abs  970335 #ECE5F -   917
=aSTATT  Abs  970205 #ECDDD -   856
=aSTLUP  Abs  971830 #ED436 -  1530  1609  1680
=aSTNIB  Abs  970940 #ED0BC -  1150  1256  1324  1366  1373
=aTRIM   Abs  971705 #ED3B9 -  1480
 zLEFT   Abs     154 #0009A -  1749  1750
 zNEXT   Abs  972544 #ED700 -     5  1749
 zSIZE   Abs    4198 #01066 -  1748
 zTHIS   Abs  968192 #EC600 -     4  1748
```



```
Input Parameters

  Source file name is GN&AS5

  Listing file name is GN/AS5::65

  Object file name is GN%AS5::65

                                    111111
                          0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News
```



```
 1                      TITLE  AS6:_FORTH_ASSEMBLER_PSEUDO-OPS
 2                      RDSYMB MR%GTO
 3 ED700                ABS    #ED700
 4              zTHIS   EQU    *
 5              zNEXT   EQU    #EEF00
 6
 7              ********************************************************
 8              ***
 9              *** MR&AS6     <840504.1037>
10              ***
11              ***
12              ***  FORTH ASSEMBLER
13              ***  PSEUDO OPS:  CONTROL, CONSTANT, AND MACRO EXPANSION
14              ***
15              ********************************************************
16              ****************************************************************
17              ***
18              ***  BSS
19              ***
20              *** ( -- [NUM] )
21              *** ( BLOCK OF ZEROS )
22              ****************************************************************
23 ED700 0000  =aBSS   CON(5) =DOCOL          : BSS
        0
24 ED705 0000          CON(5) =aEXPR          EXPR          ( * NUM * )
        0
25 ED70A 0000          CON(5) =a?PAS1         ?PASS=1
        0
26 ED70F 0000          CON(5) =ZBRNH          IF            ( * NUM * )
        0
27 ED714 F000          REL(5) %IF104
        0
28              *
29              * PASS 1 PROCESSING
30              *
31 ED719 0000          CON(5) =ABS            ABS           ( * NUM * )
        0
32 ED71E 0000          CON(5) =SEMI           ;             ( * NUM * )
        0
33              *
34              * PASS 2 PROCESSING
35              *
36 ED723        %IF104                        ELSE          ( * NUM * )
37 ED725 0000          CON(5) =DUP            DUP
        0
38 ED728 0000          CON(5) =LTZ            0<            ( * NUM F * )
        0
39 ED72D 0000          CON(5) =ZBRNH          IF            ( * NUM * )
        0
40 ED732 9100          REL(5) %IF151
        0
41              *
42              * NEGATIVE BSS GIVEN.  REPORT ERROR & TAKE ABSOLUTE VALUE
43              *
44 ED737 0000          CON(5) =ABS            ABS           ( * NUM * )
```

```
          0
 45 ED73C 0000          CON(5) =LIT             39
          0
 46 ED741 7200          CON(5) 39               'illegal expression'
          0
 47 ED746 0000          CON(5) =aERROR          ERROR       ( * NUM * )
          0
 48                *
 49                * SAVE OPCODE LENGTH
 50                *
 51 ED74B        %IF151                          THEN        ( * NUM * )
 52 ED74B 0000          CON(5) =DUP             DUP
          0
 53 ED750 0000          CON(5) =aOPLEN          OPLEN
          0
 54 ED755 0000          CON(5) =STORE           !           ( * NUM * )
          0
 55                *
 56                * PUT ZERO IN (AT MOST 18) OPCODES
 57                *
 58 ED75A 0000          CON(5) =LIT             18
          0
 59 ED75E 2100          CON(5) 18
          0
 60 ED764 0000          CON(5) =MIN             MIN
          0
 61 ED769 0000          CON(5) =ONE             1
          0
 62 ED76E 0000          CON(5) =MAX             MAX
          0
 63 ED773 0000          CON(5) =ZERO            0
          0
 64 ED778 0000          CON(5) =XDO             DO          ( * * )
          0
 65 ED77D        %DO12
 66 ED77D 0000          CON(5) =ZERO            0
          0
 67 ED782 0000          CON(5) =R@              I
          0
 68 ED787 0000          CON(5) =ONE+            1+
          0
 69 ED78C 0000          CON(5) =a>OPx           >OPx
          0
 70 ED791 0000          CON(5) =STORE           !           ( * * )
          0
 71 ED796 0000          CON(5) =XLOOP           LOOP        ( * * )
          0
 72 ED79B 2EFF          REL(5) %DO12
          F
 73                *
 74                * ALL DONE
 75                *
 76 ED7A0 0000          CON(5) =SEMI            ;           ( * * )
          0
 77                ****************************************************************
```



```
 78                ***
 79                ***  END
 80                ***
 81                *** ( -- [NUM] )
 82                *** ( END OF SOURCE )
 83                ****************************************************************
 84 ED7A5 0000 =aEND    CON(5) =DOCOL           : END
          0
 85 ED7AA 0000          CON(5) =a?PAS1          ?PASS=1
          0
 86 ED7AF 0000          CON(5) =ZBRNH           IF          ( * * )
          0
 87 ED7B4 A000          REL(5) %IF105
          0
 88                *
 89                * PASS 1 PROCESSING
 90                *
 91 ED7B9 0000          CON(5) =ZERO            0
          0
 92                *
 93                * PASS 1 & 2 PROCESSING
 94                *
 95 ED7BE        %IF105                          THEN
 96                *
 97                * SET VARIABLE DONE TRUE
 98                *
 99 ED7BE 0000          CON(5) =aNEG1           -1
          0
100 ED7C3 0000          CON(5) =aDONE           DONE
          0
101 ED7C8 0000          CON(5) =STORE           !
          0
102 ED7CD 0000          CON(5) =SEMI            ;           ( * [NUM] * )
          0
103               ****************************************************************
104               ***
105               ***  EQU
106               ***
107               *** ( -- [NUM] )
108               *** ( EQUATE A SYMBOL )
109               ****************************************************************
110 ED7D2 0000 =aEQU    CON(5) =DOCOL           : EQU
          0
111               *
112               * CHECK IF LABEL EXISTS AND IS VALID
113               *
114 ED7D7 0000          CON(5) =aLAB.F          LABEL.FIELD ( * STR * )
          0
115 ED7DC 0000          CON(5) =SWAP            SWAP
          0
116 ED7E1 0000          CON(5) =DROP            DROP        ( * NUM * )
          0
117 ED7E6 0000          CON(5) =aLAB.F          LABEL.FIELD ( * NUM STR * )
          0
118 ED7EB 0000          CON(5) =aLABOK          LABEL.OK?   ( * NUM F * )
```

```
          0
  119 ED7F0 0000        CON(5) =AND           AND          ( * F * )
          0
  120 ED7F5 0000        CON(5) =ZBRNH         IF           ( * * )
          0
  121 ED7FA FF00        REL(5) %IF142
          0
  122           *
  123           * ALL OK.
  124           *
  125 ED7FF 0000        CON(5) =aLAB.F        LABEL.FIELD  ( * STR * )
          0
  126 ED804 0000        CON(5) =aLUKUP        LOOKUP       ( * PTR * )
          0
  127 ED809 0000        CON(5) =a?PAS1        ?PASS=1      ( * PTR F * )
          0
  128 ED80E 0000        CON(5) =ZBRNH         IF           ( * PTR * )
          0
  129 ED813 5500        REL(5) %IF144
          0
  130           *
  131           * PASS 1 PROCESSING
  132           *
  133 ED818 0000        CON(5) =ZERO          0            ( * PTR NUM * )
          0
  134 ED81D 0000        CON(5) =SWAP          SWAP         ( * NUM PTR * )
          0
  135 ED822 0000        CON(5) =ZEQ           0=           ( * NUM F * )
          0
  136 ED827 0000        CON(5) =ZBRNH         IF           ( * NUM * )
          0
  137 ED82C 5F00        REL(5) %IF143
          0
  138           *
  139           * NEW EQUATE.  LET'S PUT IT INTO THE SYMBOL TABLE
  140           *
  141 ED831 0000        CON(5) =aEXPR         EXPRESSION   ( * NUM NUM' * )
          0
  142 ED836 0000        CON(5) =aLAB.F        LABEL.FIELD  ( * NUM NUM' STR * )
          0
  143 ED83B 0000        CON(5) =a>SYT         >SYT         ( * NUM * )
          0
  144           *
  145           * SET TYPE NIBBLE TO ZERO (EQUATE)
  146           *
  147 ED840 0000        CON(5) =ZERO          0            ( * NUM NUM' * )
          0
  148 ED845 0000        CON(5) =aLAB.F        LABEL.FIELD  ( * NUM NUM' STR * )
          0
  149 ED84A 0000        CON(5) =aLUKUP        LOOKUP       ( * NUM NUM' PTR * )
          0
  150 ED84F 0000        CON(5) =LIT           17           ( * NUM NUM' PTR NUM' *
          0
  151 ED854 1100        CON(5) 17
          0
```



```
  152 ED859 0000        CON(5) =ADD           +            ( * NUM NUM' PTR * )
          0
  153 ED85E 0000        CON(5) =N!            N!           ( * NUM * )
          0
  154 ED863 0000        CON(5) =SEMI          ;            ( * NUM * )
          0
  155           *
  156           * PASS 2 PROCESSING
  157           *
  158 ED868       %IF144                      ELSE         ( * PTR * )
  159 ED868 0000        CON(5) =DUP           DUP          ( * PTR PTR * )
          0
  160 ED86D 0000        CON(5) =LIT           17           ( * PTR PTR 17 * )
          0
  161 ED872 1100        CON(5) 17
          0
  162 ED877 0000        CON(5) =ADD           +            ( * PTR PTR' * )
          0
  163 ED87C 0000        CON(5) =DUP           DUP          ( * PTR PTR' PTR' * )
          0
  164 ED881 0000        CON(5) =N@            N@           ( * PTR PTR' NIB * )
          0
  165 ED886 0000        CON(5) =ZBRNH         IF           ( * PTR PTR' * )
          0
  166 ED88B E100        REL(5) %IF145
          0
  167           *
  168           * ENTRY IN SYMBOL TABLE IS LABEL, NOT EQUATE.  WE HAVE CONFLICT!
  169           *
  170 ED890 0000        CON(5) =2DROP         2DROP        ( * * )
          0
  171 ED895 0000        CON(5) =LIT           41           ( * NUM * )
          0
  172 ED89A 9200        CON(5) 41             'duplicate label'
          0
  173 ED89F 0000        CON(5) =aERROR        ERROR        ( * * )
          0
  174 ED8A4 0000        CON(5) =SEMI          ;            ( * * )
          0
  175           *
  176           * ENTRY IN SYMBOL TABLE IS EQUATE.  SET TYPE NIBBLE TO 2 (EQUATE
  177           *
  178 ED8A9       %IF145                      ELSE         ( * PTR PTR' * )
  179 ED8A9 0000        CON(5) =TWO           2            ( * PTR PTR' NUM * )
          0
  180 ED8AE 0000        CON(5) =SWAP          SWAP         ( * PTR NUM PTR' * )
          0
  181 ED8B3 0000        CON(5) =N!            N!           ( * PTR * )
          0
  182 ED8B8 0000        CON(5) =LIT           12
          0
  183 ED8BD C000        CON(5) 12
          0
  184 ED8C2 0000        CON(5) =ADD           +            ( * PTR * )
          0
```

```
185 ED8C7 0000          CON(5) =AT              @           ( * NUM * )
          0
186 ED8CC 0000          CON(5) =aEXPR           EXPRESSION  ( * NUM NUM' * )
          0
187 ED8D1 0000          CON(5) =EQUAL           =           ( * F * )
          0
188 ED8D6 0000          CON(5) =ZEQ             0=          ( * F * )
          0
189 ED8DB 0000          CON(5) =ZBRNH           IF          ( * * )
          0
190 ED8E0 1400          REL(5) %IF143
          0
191             *
192             * VALUE IN SYMBOL TABLE DIFFERS FROM PASS 2 VALUE
193             *
194 ED8E5 0000          CON(5) =LIT             57          ( * NUM * )
          0
195 ED8EA 9300          CON(5) 57               'cannot resolve equate'
          0
196 ED8EF 0000          CON(5) =aERROR          ERROR       ( * * )
          0
197 ED8F4 0000          CON(5) =SEMI            ;           ( * * )
          0
198             *
199             * INVALID OR MISSING LABEL
200             *
201 ED8F9        %IF142                         ELSE        ( * * )
202 ED8F9 0000          CON(5) =a?PAS1          ?PASS=1     ( * F * )
          0
203 ED8FE 0000          CON(5) =ZBRNH           IF          ( * * )
          0
204 ED903 F000          REL(5) %IF146
          0
205 ED908 0000          CON(5) =ZERO            0           ( * NUM * )
          0
206 ED90D 0000          CON(5) =SEMI            ;           ( * NUM * )
          0
207             *
208             * PASS 2 - REPORT ERROR
209             *
210 ED912        %IF146                         ELSE        ( * * )
211 ED912 0000          CON(5) =LIT             22          ( * NUM * )
          0
212 ED917 6100          CON(5) 22               'illegal/missing label'
          0
213 ED91C 0000          CON(5) =aERROR          ERROR       ( * * )
          0
214             *
215             * ALL DONE
216             *
217 ED921        %IF143
218 ED921 0000          CON(5) =SEMI            ;           ( * [NUM] * )
          0
219             ****************************************************************
220             ***
```



```
221             *** LIST
222             ***
223             *** ( -- )
224             *** ( TURN ON AND OFF THE LISTING )
225             ****************************************************************
226 ED926 0000 =aLIST  CON(5) =DOCOL          : LIST
          0
227 ED92B 0000          CON(5) =aEXP.F        EXPRESSION  ( * STR * )
          0
228 ED930 0000          CON(5) =2DUP          2DUP
          0
229 ED935 0000          CON(5) =QUOTC         " ON"       ( * STR STR STR' * )
          0
230 ED93A 20            CON(2) 2
231 ED93C 20            CON(2) 2
232 ED93E F4E4          NIBASC \ON\
233 ED942 0000          CON(5) =S=            S=          ( * STR F * )
          0
234 ED947 0000          CON(5) =ZBRNH         IF          ( * STR * )
          0
235 ED94C E100          REL(5) %IF107
          0
236             *
237             * EXPRESSION FIELD IS ON.  SET VARIABLE TOLIST TRUE
238             *
239 ED951 0000          CON(5) =2DROP         2DROP       ( * * )
          0
240 ED956 0000          CON(5) =aNEG1         -1
          0
241 ED95B 0000          CON(5) =aTOLST        TOLIST
          0
242 ED960 0000          CON(5) =STORE         !           ( * * )
          0
243 ED965 0000          CON(5) =SEMI          ;           ( * * )
          0
244 ED96A        %IF107                       ELSE        ( * STR * )
245 ED96A 0000          CON(5) =QUOTC         "           ( * STR STR' * )
          0
246 ED96F 30            CON(2) 3
247 ED971 30            CON(2) 3
248 ED973 F464          NIBASC \OFF\
          64
249 ED979 0000          CON(5) =S=            S=          ( * F * )
          0
250 ED97E 0000          CON(5) =ZBRNH         IF          ( * * )
          0
251 ED983 9100          REL(5) %IF108
          0
252             *
253             * EXPRESSION FIELD IS OFF.  SET VARIABLE TOLIST FALSE
254             *
255 ED988 0000          CON(5) =ZERO          0
          0
256 ED98D 0000          CON(5) =aTOLST        TOLIST
          0
```

```
257 ED992 0000          CON(5) =STORE          !           ( * * )
          0
258 ED997 0000          CON(5) =SEMI           ;           ( * * )
          0
259             *
260             * EXPRESSION FIELD IS NEITHER ON NOR OFF
261             *
262 ED99C        %IF108
263 ED99C 0000          CON(5) =LIT            23
          0
264 ED9A1 7100          CON(5) 23              'invalid listing arguement'
          0
265 ED9A6 0000          CON(5) =aERROR         ERROR       ( * * )
          0
266 ED9AB 0000          CON(5) =SEMI           ;           ( * * )
          0
267             ****************************************************************
268             ***
269             ***  SKIP LABEL
270             ***
271             *** ( STR -- STR )
272             *** ( SKIP OVER CHARACTERS IN STRING EQUAL TO THE CHARACTERS IN T
273             *** ( LABEL FIELD )
274             ****************************************************************
275 ED9B0 0000 =a+LAB  CON(5) =DOCOL          : SKIP LABEL
          0
276             *
277             * GET LENGTH OF LABEL FIELD
278             *
279 ED9B5 0000          CON(5) =SWAP           SWAP        ( * LEN ADDR * )
          0
280 ED9BA 0000          CON(5) =aLAB.F         LABEL.FIELD ( * LEN ADDR STR * )
          0
281 ED9BF 0000          CON(5) =SWAP           SWAP
          0
282 ED9C4 0000          CON(5) =DROP           DROP        ( * LEN ADDR LEN' * )
          0
283             *
284             * UPDATE ADDRESS BY CHARACTERS*2
285             *
286 ED9C9 0000          CON(5) =DUP            DUP
          0
287 ED9CE 0000          CON(5) =>R             >R
          0
288 ED9D3 0000          CON(5) =TWO*           2*          ( * LEN ADDR CNT * )
          0
289 ED9D8 0000          CON(5) =ADD            +           ( * LEN ADDR' * )
          0
290 ED9DD 0000          CON(5) =SWAP           SWAP        ( * STR * )
          0
291             *
292             * UPDATE LENGTH BY CHARACTERS
293             *
294 ED9E2 0000          CON(5) =R>             R>
          0
```



```
295 ED9E7 0000          CON(5) =MINUS          -           ( * STR * )
          0
296 ED9EC 0000          CON(5) =SEMI           ;           ( * STR * )
          0
297             ****************************************************************
298             ***
299             ***  EXTRACT
300             ***
301             *** ( -- STR )
302             *** ( SUCK STITLE OR TITLE STRING FROM SOURCE LINE )
303             ****************************************************************
304 ED9F1 0000 =aXTRCT CON(5) =DOCOL          : EXTRACT
          0
305             *
306             * FIND TITLE/STITLE PSEUDO-OP IN SOURCE LINE
307             *
308 ED9F6 0000          CON(5) =QUOTC          " TITLE"    ( * STR * )
          0
309 ED9FB 50            CON(2) 5
310 ED9FD 50            CON(2) 5
311 ED9FF 4594          NIBASC \TITLE\
          45C4
          54
312 EDA09 0000          CON(5) =aSLINE         SLINE
          0
313 EDA0E 0B9D          CON(5) =a+LAB          SKIP LABEL  ( * STR STR' * )
          E
314 EDA13 0000          CON(5) =POS            POS         ( * NUM * )
          0
315             *
316             * EXTRACT TEXT FOLLOWING TITLE OR STITLE PSEUDO-OP
317             *
318 EDA18 0000          CON(5) =?DUP           ?DUP
          0
319 EDA1D 0000          CON(5) =ZBRNH          IF          ( * [NUM] * )
          0
320 EDA22 E600          REL(5) %IF109
          0
321 EDA27 0000          CON(5) =FIVE+          5+
          0
322 EDA2C 0000          CON(5) =aSLINE         SLINE       ( * COL STR * )
          0
323 EDA31 0B9D          CON(5) =a+LAB          SKIP LABEL
          E
324 EDA36 0000          CON(5) =ROT            ROT
          0
325 EDA3B 0000          CON(5) =END$           END$        ( * STR * )
          0
326             *
327             * TRIM OFF LEADING BLANKS
328             *
329 EDA40        %BG13                         BEGIN       ( * STR * )
330 EDA40 0000          CON(5) =OVER           OVER
          0
331 EDA45 0000          CON(5) =CAT            C@
```

```
          0
   332 EDA4A 0000         CON(5) =BL            BLANK
          0
   333 EDA4F 0000         CON(5) =EQUAL         =           ( * STR F * )
          0
   334 EDA54 0000         CON(5) =OVER          OVER
          0
   335 EDA59 0000         CON(5) =F.AND         F.AND       ( * STR F * )
          0
   336 EDA5E 0000         CON(5) =ZBRNH         WHILE
          0
   337 EDA63 9100         REL(5) %RP13
          0
   338 EDA68 0000         CON(5) =C@+           C@+
          0
   339 EDA6D 0000         CON(5) =DROP          DROP        ( * STR * )
          0
   340 EDA72 0000         CON(5) =BRNCH         REPEAT
          0
   341 EDA77 9CFF         REL(5) %BG13
          F
   342 EDA7C       %RP13
   343             *
   344             * TRIM TO CORRECT LENGTH
   345             *
   346 EDA7C 0000         CON(5) =LIT           40
          0
   347 EDA81 8200         CON(5) 40
          0
   348 EDA86 0000         CON(5) =MIN           MIN         ( * STR * )
          0
   349 EDA8B 0000         CON(5) =SEMI          ;           ( * STR * )
          0
   350 EDA90       %IF109
   351 EDA90 0000         CON(5) =aBLNK1        1BLANK
          0
   352 EDA95 0000         CON(5) =SEMI          ;           ( * STR * )
          0
   353             ****************************************************************
   354             ***
   355             ***  STITLE
   356             ***
   357             *** ( -- )
   358             *** ( SET THE SUBTITLE LINE )
   359             ****************************************************************
   360 EDA9A 0000 =aSTITL CON(5) =DOCOL         : STITLE
          0
   361 EDA9F 1F9D         CON(5) =aXTRCT        EXTRACT
          E
   362 EDAA4 0000         CON(5) =aSTIT         STIT
          0
   363 EDAA9 0000         CON(5) =S!            S!
          0
   364 EDAAE 0000         CON(5) =aEJECT        EJECT
          0
```



```
   365 EDAB3 0000         CON(5) =SEMI          ;           ( * * )
          0
   366             ****************************************************************
   367             ***
   368             ***  TITLE
   369             ***
   370             *** ( -- )
   371             *** ( SET THE TITLE LINE )
   372             ****************************************************************
   373 EDAB8 0000 =aTITLE CON(5) =DOCOL         : TITLE
          0
   374 EDABD 1F9D         CON(5) =aXTRCT        EXTRACT
          E
   375 EDAC2 0000         CON(5) =aTIT          TIT
          0
   376 EDAC7 0000         CON(5) =S!            S!
          0
   377 EDACC 0000         CON(5) =SEMI          ;           ( * * )
          0
   378             ****************************************************************
   379             ***
   380             ***  SIM.LINE
   381             ***
   382             *** ( STR -- )
   383             *** ( PASS A SIMULATED LINE TO THE ASSEMBLER )
   384             ****************************************************************
   385 EDAD1 0000 =aSIMLN CON(5) =DOCOL         : SIM.LINE
          0
   386             *
   387             * SAVE STRING ON STACK IN VARIABLE SOURCE LINE
   388             *
   389 EDAD6 0000         CON(5) =aSLINE        SLINE
          0
   390 EDADB 0000         CON(5) =S!            S!
          0
   391             *
   392             * CALL HIGH LEVEL ROUTINE TO PROCESS ONE LINE
   393             *
   394 EDAE0 0000         CON(5) =aDOLIN        DO.LINE
          0
   395             *
   396             * RESET VARIABLE LLINE
   397             *
   398 EDAE5 0000         CON(5) =a&CLR         &CLEAR
          0
   399 EDAEA 0000         CON(5) =SEMI          ;           ( * * )
          0
   400             ****************************************************************
   401             ***
   402             ***  WRAPUPOLD
   403             ***
   404             *** ( -- )
   405             *** ( SAVE CURRENT EXPRESSION & FINISH PROCESSING LINE )
   406             ****************************************************************
   407 EDAEF 0000 =aWRUOL CON(5) =DOCOL         : WRAPUPOLD
```

```
         0
408              *
409              * CURRENT LINE IS A MACRO-EXPANSION PSEUDO-OP, WHICH GENERATES
410              * NO CODE ON THIS LINE, BUT IT'S NOT DONE.  THE FULL STEPS FOR
411              * PROCESSING A LINE ARE:
412              *   PARSE - PROCESS - SPIT - LC+
413              * SINCE WE'RE IN THE MIDDLE OF PROCESSING, AND WE WANT TO WRAP
414              * UP THIS LINE, BUT CONTINUE, WE ONLY DO THE LATTER PARTS.
415              *
416              *
417              * GENERATE NO CODE THIS LINE
418              *
419 EDAF4 0000            CON(5) =ZERO          0
         0
420 EDAF9 0000            CON(5) =aOPLEN        OPLEN
         0
421 EDAFE 0000            CON(5) =STORE         !           ( * * )
         0
422              *
423              * SAVE THIS EXPRESSION FIELD, JUST ABOUT EVERY ROUTINE USES IT.
424              *
425 EDB03 0000            CON(5) =aEXP.F        EXPRESSION  ( * STR * )
         0
426 EDB08 0000            CON(5) =aOLDEX        OLDEXPR
         0
427 EDB0D 0000            CON(5) =S!            S!          ( * * )
         0
428              *
429              * SPIT
430              *
431 EDB12 0000            CON(5) =aSPIT         SPIT
         0
432              *
433              * LC+
434              *
435 EDB17 0000            CON(5) =aLC+          LC+
         0
436              *
437              * CLEAR VARIABLE LLINE
438              *
439 EDB1C 0000            CON(5) =a&CLR         &CLEAR
         0
440 EDB21 0000            CON(5) =SEMI          ;           ( * * )
         0
441              ***************************************************************
442              ***
443              ***  STARTUPNEW
444              ***
445              *** ( STR -- )
446              *** ( USE STR TO START A NEW LINE )
447              ***************************************************************
448 EDB26 0000 =aSTUNW CON(5) =DOCOL         : STARTUPNEW
         0
449              *
450              * START UP NEW, LIKE WRAP UP OLD, IS CALLED WHEN MACRO-EXPANSION
```



```
451              * ARE ENCOUNTERD.  WHEN WE EXIT BACK TO WHAT CALLED US, DOLINE WI
452              * WITH SPIT & LC+.  SO FOR THE LINE WE'RE SIMULATING, WE ONLY WAN
453              * THE FIRST TWO STEPS:  PARSE & PROCESS
454              *
455 EDB2B 0000            CON(5) =aSLINE        SLINE
         0
456 EDB30 0000            CON(5) =S!            S!
         0
457 EDB35 0000            CON(5) =aPARSE        PARSE
         0
458 EDB3A 0000            CON(5) =aPROCS        PROCESS
         0
459 EDB3F 0000            CON(5) =SEMI          ;           ( * * )
         0
460              ***************************************************************
461              ***
462              ***  ?1ST TIME THROUGH
463              ***
464              *** ( -- F )
465              *** ( CHECK TO SEE IF FILETYPE ALREADY DECLARED )
466              ***************************************************************
467 EDB44 0000 =a?1ST  CON(5) =DOCOL         : ?1ST TIME THROUGH
         0
468 EDB49 0000            CON(5) =a?PAS1        ?PASS=1
         0
469 EDB4E 0000            CON(5) =ZEQ           0=          ( * F * )
         0
470 EDB53 0000            CON(5) =ZBRNH         IF          ( * * )
         0
471 EDB58 1400            REL(5) %IF148
         0
472              *
473              * PASS 2 PROCESSING
474              *
475 EDB5D 0000            CON(5) =TWO           2
         0
476 EDB62 0000            CON(5) =aTESTF        TEST.FLAG   ( * F * )
         0
477 EDB67 0000            CON(5) =ZEQ           0=          ( * F * )
         0
478 EDB6C 0000            CON(5) =ZBRNH         IF          ( * * )
         0
479 EDB71 9100            REL(5) %IF147
         0
480              *
481              * HAVEN'T BEEN HERE BEFORE.  SET FLAG SO WE'LL RECOGNIZE IT WHEN
482              *
483 EDB76 0000            CON(5) =TWO           2           ( * NUM * )
         0
484 EDB7B 0000            CON(5) =aSETF         SET.FLAG
         0
485 EDB80 0000            CON(5) =aNEG1         -1          ( * TF * )
         0
486 EDB85 0000            CON(5) =SEMI          ;           ( * TF * )
         0
```

```
                *
                * BEEN IN THIS ROUTINE BEFORE!
                *
EDB8A            %IF147
EDB8A 0000              CON(5) =LIT             51          ( * NUM * )
      0
EDB8F 3300              CON(5) 51               'multiple file type'
      0
EDB94 0000              CON(5) =aERROR          ERROR       ( * * )
      0
                *
                * PASS 1 PROCESSING
                *
EDB99            %IF148
EDB99 0000              CON(5) =ZERO            0           ( * FF * )
      0
EDB9E 0000              CON(5) =SEMI            ;           ( * FF * )
      0
                ****************************************************************
                ***
                ***  FILE.IS.FORTH
                ***
                *** ( -- )
                *** ( 'FORTH' PSEUDO-OP FOUND. )
                *** ( IF FIRST PASS: PURGE ASMBOB & SET LC TO FORTHRAM )
                *** ( IF SECOND PASS: ALLOT SPACE, ISSUE MULTIPLE FILETYPE ERRORS
                ****************************************************************
EDBA3 0000 =aFIL4T CON(5) =DOCOL        : FILE.IS.FORTH
      0
EDBA8 0000              CON(5) =aFLTYP          FILETYPE    ( * PTR * )
      0
EDBAD 0000              CON(5) =AT              @           ( * NUM * )
      0
EDBB2 0000              CON(5) =ZEQ             0=          ( * F * )
      0
EDBB7 0000              CON(5) =ZBRNH           IF          ( * * )
      0
EDBBC 5600              REL(5) %IF149
      0
                *
                * FIRST TIME THROUGH, WE'RE A FORTH ASSEMBLY
                *
EDBC1 0000              CON(5) =ONE             1           ( * NUM * )
      0
EDBC6 0000              CON(5) =aFLTYP          FILETYPE
      0
EDBCB 0000              CON(5) =STORE           !           ( * * )
      0
                *
                * NO OBJECT FILE REQUIRED:  FORTHRAM IS OUR OBJECT FILE
                *
EDBD0 0000              CON(5) =QUOTC           " ASMBOB"
      0
EDBD5 60                CON(2) 6
EDBD7 60                CON(2) 6
```



```
EDBD9 1435              NIBASC \ASMBOB\
      D424
      F424
EDBE5 0000              CON(5) =aKILL           KILL        ( * * )
      0
EDBEA 0000              CON(5) =aMEMOV          MEMMOVE     ( * * )
      0
                *
                * MARK THE LOSS OF AN OBJECT FILE
                *
EDBEF 0000              CON(5) =NULL$           NULL$
      0
EDBF4 0000              CON(5) =aOFILE          OFILE
      0
EDBF9 0000              CON(5) =S!              S!          ( * * )
      0
                *
                * BOTH VARIABLE LC AND VARIABLE OBJFILE MUST REFLECT THIS CHANGE
                *
EDBFE 0000              CON(5) =HERE            HERE
      0
EDC03 0000              CON(5) =DUP             DUP         ( * ADDR ADDR * )
      0
EDC08 0000              CON(5) =aOBJFL          OBJFILE
      0
EDC0D 0000              CON(5) =STORE           !
      0
EDC12 0000              CON(5) =aLC             LC
      0
EDC17 0000              CON(5) =STORE           !           ( * * )
      0
EDC1C 0000              CON(5) =SEMI            ;           ( * * )
      0
                *
                * BEEN HERE BEFORE, BUT WHICH PASS?
                *
EDC21            %IF149                         ELSE        ( * * )
EDC21 44BD              CON(5) =a?1ST           ?1ST TIME   ( * F * )
      E
EDC26 0000              CON(5) =ZBRNH           IF          ( * * )
      0
EDC2B A500              REL(5) %IF150
      0
                *
                * SECOND PASS.  WE WERE HERE ONLY DURING PASS 1
                *
                *
                * CALCULATE NUMBER OF NIBBLES NEEDED IN FORTH DICTIONARY
                *
EDC30 0000              CON(5) =HERE            HERE
      0
EDC35 0000              CON(5) =S0              S0
      0
EDC3A 0000              CON(5) =LIT             458
      0
```

```
562 EDC3F AC10          CON(5) 458
        0
563 EDC44 0000          CON(5) =MINUS           -               ( * NUM NUM' * )
        0
564 EDC49 0000          CON(5) =aFLSIZ          FILESIZE
        0
565 EDC4E 0000          CON(5) =AT              @
        0
566 EDC53 0000          CON(5) =MINUS           -
        0
567 EDC58 0000          CON(5) =GT              >
        0
568 EDC5D 0000          CON(5) =ZBRNH           IF              ( * * )
        0
569 EDC62 4100          REL(5) %IF111
        0
570             *
571             * NOT ENOUGH IN DICTIONARY
572             *
573 EDC67 0000          CON(5) =LIT             8
        0
574 EDC6C 8000          CON(5) 8                'dictionary full'
        0
575 EDC71 0000          CON(5) =aSHUTD          SHUTDOWN
        0
576             *
577             * ENOUGH.  ALLOT GUARENTEED NOT TO FAIL (TEST JUST MADE)
578             *
579 EDC76        %IF111
580 EDC76 0000          CON(5) =aFLSIZ          FILESIZE
        0
581 EDC7B 0000          CON(5) =AT              @               ( * NUM * )
        0
582 EDC80 0000          CON(5) =ALLOT           NALLOT          ( * * )
        0
583             *
584             * ALL DONE
585             *
586 EDC85        %IF150
587 EDC85 0000          CON(5) =SEMI            ;               ( * * )
        0
588             ****************************************************************
589             ***
590             ***  FILE.IS.LEX
591             ***
592             *** ( -- )
593             *** ( 'LEX' PSEUDO-OP FOUND.  DO NEEDED PREP )
594             ****************************************************************
595 EDC8A 0000 =aFILLX CON(5) =DOCOL           : FILE.IS.LEX
        0
596 EDC8F 0000          CON(5) =aFLTYP          FILETYPE
        0
597 EDC94 0000          CON(5) =AT              @               ( * NUM * )
        0
598 EDC99 0000          CON(5) =ZBRNH           IF              ( * * )
```



```
        0
599 EDC9E 0000          CON(5) =ZBRNH           IF              ( * * )
        0
600 EDCA3 B400          REL(5) %IF115
        0
601             *
602             * FIRST TIME THROUGH, WE'RE A LEX ASSEMBLY
603             *
604 EDCA8 0000          CON(5) =TWO             2
        0
605 EDCAD 0000          CON(5) =aFLTYP          FILETYPE
        0
606 EDCB2 0000          CON(5) =STORE           !               ( * * )
        0
607 EDCB7 0000          CON(5) =SEMI            ;               ( * * )
        0
608             ****************************************************************
609             ***
610             ***  FILE.IS.BIN
611             ***
612             *** ( -- )
613             *** ( 'BIN' PSEUDO-OP FOUND.  DO NEEDED PREP )
614             ****************************************************************
615 EDCBC 0000 =aFILBN CON(5) =DOCOL           : FILE.IS.BIN
        0
616 EDCC1 0000          CON(5) =aFLTYP          FILETYPE
        0
617 EDCC6 0000          CON(5) =AT              @               ( * NUM * )
        0
618 EDCCB 0000          CON(5) =ZEQ             0=
        0
619 EDCD0 0000          CON(5) =ZBRNH           IF              ( * * )
        0
620 EDCD5 9100          REL(5) %IF115
        0
621             *
622             * FIRST TIME THROUGH, WE'RE A BIN ASSEMBLY
623             *
624 EDCDA 0000          CON(5) =THREE           3
        0
625 EDCDF 0000          CON(5) =aFLTYP          FILETYPE
        0
626 EDCE4 0000          CON(5) =STORE           !               ( * * )
        0
627 EDCE9 0000          CON(5) =SEMI            ;               ( * * )
        0
628             *
629             * SECOND TIME THROUGH.  IS IT BECAUSE THE FIRST TIME THROUGH WAS
630             * PASS 1 ?
631             *
632 EDCEE        %IF115                          ELSE
633 EDCEE 44BD          CON(5) =a?1ST           ?1ST TRY        ( * F * )
        E
634 EDCF3 0000          CON(5) =ZBRNH           IF              ( * * )
        0
```

```
635 EDCF8 A000          REL(5) %IF156
          0
636             *
637             * YES.  WE NEED TO GROW THE OBJECT FILE TO INCLUDE THE NIBBLES
638             * WE'LL GENERATE
639             *
640 EDCFD 0000          CON(5) =aOBJGR          OBJ.GROW
          0
641             *
642             * DONE PROCESSING BIN OR LEX PSEUDO-OP
643             *
644 EDD02        %IF156                         THEN        ( * * )
645 EDD02 FEAD          CON(5) =aWRUOL          WRAPUPOLD   ( * * )
          E
646 EDD07 0000          CON(5) =SEMI            ;           ( * * )
          0
647             ****************************************************************
648             ***
649             ***  FILE.TEST
650             ***
651             *** ( NUM -- )
652             *** ( ENSURE FILE TYPE MATCHES 'NUM' )
653             ****************************************************************
654 EDD0C 0000 =aFLTST CON(5) =DOCOL          : FILE.TEST
          0
655 EDD11 0000          CON(5) =aFLTYP          FILETYPE
          0
656 EDD16 0000          CON(5) =AT              @           ( * NUM NUM' * )
          0
657 EDD1B 0000          CON(5) =EQUAL           =
          0
658 EDD20 0000          CON(5) =ZEQ             0=          ( * F * )
          0
659 EDD25 0000          CON(5) =ZBRNH           IF          ( * * )
          0
660 EDD2A 4100          REL(5) %IF112
          0
661             *
662             * PSEUDO-OP IN WRONG FILE (I.E. WORD1 IN A BIN FILE)
663             *
664 EDD2F 0000          CON(5) =LIT             51
          0
665 EDD34 3300          CON(5) 51               'missing/multiple filetype'
          0
666 EDD39 0000          CON(5) =aERROR          ERROR       ( * * )
          0
667 EDD3E        %IF112                         THEN
668 EDD3E 0000          CON(5) =SEMI            ;           ( * * )
          0
669             ****************************************************************
670             ***
671             ***  ENSURE.FORTH
672             ***
673             *** ( -- )
674             *** ( A FORTH SUBHEADER PSEUDO-OP FOUND )
```



```
675             ****************************************************************
676 EDD43 0000 =aENS4T CON(5) =DOCOL          : ENSURE.FORTH
          0
677 EDD48 0000          CON(5) =ONE             1           ( * NUM * )
          0
678 EDD4D C0DD          CON(5) =aFLTST          FILE.TEST
          E
679 EDD52 0000          CON(5) =SEMI            ;           ( * * )
          0
680             ****************************************************************
681             ***
682             ***  ENSURE.LEX
683             ***
684             *** ( -- )
685             *** ( A LEX SUBHEADER PSEUDO-OP FOUND )
686             ****************************************************************
687 EDD57 0000 =aENSLX CON(5) =DOCOL          : ENSURE.LEX
          0
688 EDD5C 0000          CON(5) =TWO             2
          0
689 EDD61 C0DD          CON(5) =aFLTST          FILE.TEST
          E
690 EDD66 0000          CON(5) =SEMI            ;           ( * * )
          0
691             ****************************************************************
692             ***
693             ***  ENSURE.BIN
694             ***
695             *** ( -- )
696             ****************************************************************
697 EDD6B 0000 =aENSBN CON(5) =DOCOL          : ENSURE.BIN
          0
698 EDD70 0000          CON(5) =THREE           3
          0
699 EDD75 C0DD          CON(5) =aFLTST          FILE.TEST
          E
700 EDD7A 0000          CON(5) =SEMI            ;           ( * * )
          0
701             ****************************************************************
702             ***
703             ***  CALCLENWORD
704             ***
705             *** ( -- NUM )
706             *** ( CALCULATE THE LENGTH OF THE WORD EXPANSION )
707             ****************************************************************
708 EDD7F 0000 =aCLCWD CON(5) =DOCOL          : CALCLENWORD
          0
709             *
710             * GET WORD
711             *
712 EDD84 0000          CON(5) =aEXP.F          EXPR        ( * STR * )
          0
713 EDD89 0000          CON(5) =aQ"STR          ".STRING    ( * STR * )
          0
714 EDD8E 0000          CON(5) =aTRIM           TRIM.BLANKS ( * STR * )
```

```
          0
715             *
716             * TRIM TO MAXIMUM LENGTH
717             *
718 EDD93 0000          CON(5) =WIDTH          WIDTH
          0
719 EDD98 0000          CON(5) =AT             @
          0
720 EDD9D 0000          CON(5) =MIN            MIN          ( * STR * )
          0
721 EDDA2 0000          CON(5) =SWAP           SWAP
          0
722 EDDA7 0000          CON(5) =DROP           DROP         ( * LEN * )
          0
723             *
724             * CHECK FOR NULL WORD
725             *
726 EDDAC 0000          CON(5) =DUP            DUP          ( * NUM NUM * )
          0
727 EDDB1 0000          CON(5) =ZEQ            0=           ( * NUM F * )
          0
728 EDDB6 0000          CON(5) =ZBRNH          IF           ( * NUM * )
          0
729 EDDBB 4100          REL(5) %IF106
          0
730 EDDC0 0000          CON(5) =LIT            11
          0
731 EDDC5 B000          CON(5) 11              'attempted to redefine null'
          0
732 EDDCA 0000          CON(5) =aSHUTD         SHUTDOWN
          0
733             *
734             * CALCULATE LENGTH & ADD IN LINK FIELD (5), LENGTH BYTE (2),
735             * AND CODE FIELD (5)
736             *
737 EDDCF          %IF106                       THEN         ( * NUM * )
738 EDDCF 0000          CON(5) =TWO*           2*
          0
739 EDDD4 0000          CON(5) =LIT            12
          0
740 EDDD9 C000          CON(5) 5+2+5
          0
741 EDDDE 0000          CON(5) =ADD            +
          0
742 EDDE3 0000          CON(5) =SEMI           ;            ( * NUM * )
          0
743             ***************************************************************
744             ***
745             ***  UPTOLFA
746             ***
747             *** ( -- )
748             *** ( GENERATE CODE THROUGH THE LINK FIELD )
749             ***************************************************************
750 EDDE8 0000 =aUTLFA CON(5) =DOCOL          : UPTOLFA
          0
```



```
751 EDDED FEAD          CON(5) =aWRUOL         WRAPUPOLD    ( * * )
          E
752 EDDF2 0000          CON(5) =QUOTC          "    CON(5) #"
          0
753 EDDF7 A0            CON(2) 10
754 EDDF9 A0            CON(2) 10
755 EDDFB 0202          NIBASC \  \
756 EDDFF 34F4          NIBASC \CON(5) #\
          E482
          5392
          0232
757 EDE0F 0000          CON(5) =a>PAD          >PAD         ( * STR * )
          0
758             *
759             * FORTH VARIABLE LATEST HAS LINK TO LAST WORD
760             *
761 EDE14 0000          CON(5) =LATE           LATEST
          0
762 EDE19 0000          CON(5) =HEX            HEX
          0
763 EDE1E 0000          CON(5) =aNUM#5         #5           ( * STR STR' * )
          0
764 EDE23 0000          CON(5) =DEC            DECIMAL
          0
765 EDE28 0000          CON(5) =S<&            S<&          ( * STR * )
          0
766             *
767             * SIMULATE LINK FIELD LINE
768             *
769 EDE2D 1DAD          CON(5) =aSIMLN         SIM.LINE     ( * * )
          E
770 EDE32 0000          CON(5) =SEMI           ;            ( * * )
          0
771             ***************************************************************
772             ***
773             ***  BREAKUP8
774             ***
775             *** ( STR -- STR1 ... STRN N )
776             *** ( BREAK UP THE STRING INTO N STRINGS OF LENGTH <= 8 )
777             ***************************************************************
778 EDE37 0000 =aBRK8  CON(5) =DOCOL          : BREAKUP8
          0
779             *
780             * INITIALIZE STRING COUNT TO ONE (1)
781             *
782 EDE3C 0000          CON(5) =ONE            1
          0
783             *
784             * repeat
785             *
786 EDE41          %BG14                       BEGIN        ( * STR N * )
787 EDE41 0000          CON(5) =OVER           OVER
          0
788 EDE46 0000          CON(5) =LIT            9
          0
```

```
 789 EDE4B 9000         CON(5) 9
           0
 790 EDE50 0000         CON(5) =LT              <
           0
 791 EDE55 0000         CON(5) =ZBRNH           IF
           0
 792 EDE5A 4100         REL(5) %IF117
           0
 793             *
 794             * MOST CURRENT STRING HAS LENGTH < 9.  WE'RE DONE.
 795             *
 796 EDE5F 0000         CON(5) =aNEG1           -1          ( * STR1 ... STRn n TF
           0
 797 EDE64 0000         CON(5) =BRNCH
           0
 798 EDE69 2800         REL(5) %TH117
           0
 799 EDE6E       %IF117                         ELSE        ( * STR N * )
 800             *
 801             * ONE MORE STRING TO COUNT
 802             *
 803 EDE6E 0000         CON(5) =ONE+            1+
           0
 804 EDE73 0000         CON(5) =>R              >R          ( * STR * )
           0
 805             *
 806             * CALCULATE LENGTH OF THIS STRING
 807             *
 808 EDE78 0000         CON(5) =DUP             DUP
           0
 809 EDE7D 0000         CON(5) =LIT             8
           0
 810 EDE82 8000         CON(5) 8
           0
 811 EDE87 0000         CON(5) =MOD             MOD
           0
 812 EDE8C 0000         CON(5) =DUP             DUP
           0
 813 EDE91 0000         CON(5) =ZEQ             0=
           0
 814 EDE96 0000         CON(5) =ZBRNH           IF          ( * STR NUM * )
           0
 815 EDE9B 4100         REL(5) %IF118
           0
 816             *
 817             * WORKING STRING HAS LENGTH DIVISIBLE BY 8.  SUBSTITUTE 8 FOR 0
 818             *
 819 EDEA0 0000         CON(5) =DROP            DROP
           0
 820 EDEA5 0000         CON(5) =LIT             8
           0
 821 EDEAA 8000         CON(5) 8
           0
 822 EDEAF       %IF118                         THEN        ( * STR NUM * )
 823             *
```



```
 824             * CALCULATE LENGTH AND SAVE ON RETURN STACK
 825             *
 826 EDEAF 0000         CON(5) =SWAP            SWAP
           0
 827 EDEB4 0000         CON(5) =OVER            OVER
           0
 828 EDEB9 0000         CON(5) =MINUS           -
           0
 829 EDEBE 0000         CON(5) =SWAP            SWAP
           0
 830 EDEC3 0000         CON(5) =>R              >R          ( * STR * )
           0
 831             *
 832             * CALCULATE ADDRESS OF NEW STRING
 833             *
 834 EDEC8 0000         CON(5) =2DUP            2DUP
           0
 835 EDECD 0000         CON(5) =TWO*            2*
           0
 836 EDED2 0000         CON(5) =ADD             +
           0
 837 EDED7 0000         CON(5) =R>              R>          ( * STR1 STR2 * )
           0
 838 EDEDC 0000         CON(5) =SWAP2           2SWAP
           0
 839 EDEE1 0000         CON(5) =R>              R>
           0
 840             *
 841             * WORKING STRING STILL NEEDS TO BE CHOPPED
 842             *
 843 EDEE6 0000         CON(5) =ZERO            0           ( * STR1 STR2 n FF * )
           0
 844             *
 845             * until <REMAINING STRING BROKEN UP>
 846             *
 847 EDEEB       %TH117                         THEN
 848 EDEEB 0000         CON(5) =ZBRNH           UNTIL
           0
 849 EDEF0 15FF         REL(5) %BG14
           F
 850             *
 851             * ALL DONE.
 852             *
 853 EDEF5 0000         CON(5) =SEMI            ;           ( * STR1 ... STRn n * )
           0
 854             ********************************************************************
 855             ***
 856             ***  ENDQUOTE
 857             ***
 858             *** ( STR -- STR )
 859             *** ( PUT A FINAL ' ON THE STRING )
 860             ********************************************************************
 861 EDEFA 0000 =aENDQ  CON(5) =DOCOL          : ENDQUOTE
           0
 862 EDEFF 0000         CON(5) =QUOTC           "
```

```
         0
863 EDF04 10          CON(2) 1
864 EDF06 10          CON(2) 1
865 EDF08 72          NIBASC \'\
866 EDF0A 0000        CON(5) =S<&           S<&
         0
867 EDF0F 0000        CON(5) =SEMI          ;             ( * STR * )
         0
868           ***************************************************************
869           ***
870           ***  SIM.ASC
871           ***
872           *** ( STR -- )
873           *** ( PUT STR IN A NIBASC STATEMENT AND ASSEMBLE IT )
874           ***************************************************************
875 EDF14 0000 =aSIMFF CON(5) =DOCOL         : SIM.ASC
         0
876           *
877           * CLEAR SCRATCH SPACE
878           *
879 EDF19 0000        CON(5) =a&CLR         &CLEAR
         0
880 EDF1E 0000        CON(5) =aLLINE        LLINE       ( * STR STR' * )
         0
881           *
882           * BUILD FIRST PART (  NIBASC ')
883           *
884 EDF23 0000        CON(5) =QUOTC         "   NIBASC '"
         0
885 EDF28 A0          CON(2) 10
886 EDF2A A0          CON(2) 10
887 EDF2C 0202        NIBASC \  \
888 EDF30 E494        NIBASC \NIBASC '\
         2414
         3534
         0272
889           *
890           * APPEND FIRST PART TO SCRATCH SPACE
891           *
892 EDF40 0000        CON(5) =S<&           S<&         ( * STR STR' * )
         0
893           *
894           * APPEND STRING ON STACK TO SCRATCH SPACE
895           *
896 EDF45 0000        CON(5) =SWAP2         2SWAP
         0
897 EDF4A 0000        CON(5) =S<&           S<&         ( * STR * )
         0
898           *
899           * ADD A TRAILING SINGLE QUOTE (')
900           *
901 EDF4F AFED        CON(5) =aENDQ         ENDQUOTE
         E
902           *
903           * SIMULATE LINE, I.E. PASS TO DOLINE
```



```
904           *
905 EDF54 1DAD        CON(5) =aSIMLN        SIM.LINE
         E
906 EDF59 0000        CON(5) =SEMI          ;             ( * * )
         0
907           ***************************************************************
908           ***
909           ***  UPTONFA
910           ***
911           *** ( STR -- )
912           *** ( GENERATE CODE THROUGH NAME FIELD )
913           ***************************************************************
914 EDF5E 0000 =aUTNFA CON(5) =DOCOL         : UPTONFA
         0
915           *
916           * SAVE NORMAL/IMMEDIATE EXPANSION LINE
917           *
918 EDF63 0000        CON(5) =aSLINE        SLINE
         0
919 EDF68 0000        CON(5) =S!            S!            ( * * )
         0
920           *
921           * GENERATE CORRECT WORD
922           *
923 EDF6D 0000        CON(5) =aOLDEX        OLDEXPR     ( * STR * )
         0
924 EDF72 0000        CON(5) =aQ"STR        ".STRING    ( * STR * )
         0
925 EDF77 0000        CON(5) =aTRIM         TRIM.BLANKS ( * STR * )
         0
926 EDF7C 0000        CON(5) =WIDTH         WIDTH
         0
927 EDF81 0000        CON(5) =AT            @
         0
928 EDF86 0000        CON(5) =MIN           MIN         ( * STR * )
         0
929 EDF8B 0000        CON(5) =aOLDEX        OLDEXPR
         0
930 EDF90 0000        CON(5) =S!            S!          ( * STR * )
         0
931           *
932           * UPDATE CURRENT TO THIS WORD
933           *
934 EDF95 0000        CON(5) =aFLTYP        FILETYPE
         0
935 EDF9A 0000        CON(5) =AT            @
         0
936 EDF9F 0000        CON(5) =ONE           1
         0
937 EDFA4 0000        CON(5) =EQUAL         =
         0
938 EDFA9 0000        CON(5) =ZBRNH         IF          ( * * )
         0
939 EDFAE 2300        REL(5) %IF119
         0
```

```
940                *
941                * SET CONTEXT TO PREVIOUS WORD
942                *
943 EDFB3 0000             CON(5) =CURR          CURRENT
          0
944 EDFB8 0000             CON(5) =AT            @
          0
945 EDFBD 0000             CON(5) =CONTX         CONTEXT
          0
946 EDFC2 0000             CON(5) =STORE         !           ( * * )
          0
947                *
948                * SET CURRENT TO THIS WORD
949                *
950 EDFC7 0000             CON(5) =aLC           LC
          0
951 EDFCC 0000             CON(5) =AT            @
          0
952 EDFD1 0000             CON(5) =CURR          CURRENT
          0
953 EDFD6 0000             CON(5) =AT            @
          0
954 EDFDB 0000             CON(5) =STORE         !           ( * * )
          0
955 EDFE0            %IF119                       THEN
956                *
957                * CHECK IF WORD UNIQUE
958                *
959 EDFE0 0000             CON(5) =aOLDEX        OLDEXPR     ( * STR * )
          0
960 EDFE5 0000             CON(5) =DROP          DROP        ( * ADDR * )
          0
961 EDFEA 0000             CON(5) =TWO-          2-
          0
962 EDFEF 0000             CON(5) =CONTX         CONTEXT
          0
963 EDFF4 0000             CON(5) =AT            @
          0
964 EDFF9 0000             CON(5) =AT            @           ( * ADDR ADDR' * )
          0
965 EDFFE 0000             CON(5) =FIND>         <FIND>
          0
966 EE003 0000             CON(5) =ZBRNH         IF          ( * [CFA B] * )
          0
967 EE008 9100             REL(5) %IF121
          0
968 EE00D 0000             CON(5) =2DROP         2DROP
          0
969 EE012 0000             CON(5) =LIT           25
          0
970 EE017 9100             CON(5) 25             'warning: word not unique'
          0
971 EE01C 0000             CON(5) =aERROR        ERROR
          0
972 EE021            %IF121                       THEN        ( * * )
```



```
973                *
974                * GENERATE LENGTH LINE
975                *
976 EE021 0000             CON(5) =aSLINE        SLINE       ( * STR * )
          0
977 EE026 0000             CON(5) =aOLDEX        OLDEXPR
          0
978 EE02B 0000             CON(5) =SWAP          SWAP
          0
979 EE030 0000             CON(5) =DROP          DROP        ( * STR LEN * )
          0
980 EE035 0000             CON(5) =aNUM#3        #3
          0
981 EE03A 0000             CON(5) =S<&           S<&         ( * STR * )
          0
982 EE03F 1DAD             CON(5) =aSIMLN        SIM.LINE    ( * * )
          E
983                *
984                * GENERATE WORD TEXT IN NIBASC STATEMENTS
985                *
986 EE044 0000             CON(5) =aOLDEX        OLDEXPR     ( * STR * )
          0
987 EE049 73ED             CON(5) =aBRK8         BREAKUP8    ( * STR1 ... STRn N * )
          E
988 EE04E 0000             CON(5) =ONE-          1-
          0
989 EE053 0000             CON(5) =?DUP          ?DUP
          0
990 EE058 0000             CON(5) =ZBRNH         IF
          0
991 EE05D E100             REL(5) %IF122                     ( * STR1 ... STRn [N] *
          0
992 EE062 0000             CON(5) =ZERO          0
          0
993 EE067 0000             CON(5) =XDO           DO
          0
994 EE06C            %DO13
995 EE06C 41FD             CON(5) =aSIMFF        SIM.ASC     ( * STR * )
          E
996 EE071 0000             CON(5) =XLOOP         LOOP
          0
997 EE076 6FFF             REL(5) %DO13
          F
998 EE07B            %IF122                       THEN        ( * STR * )
999                *
1000               * GENERATE LAST NIBASC
1001               *
1002 EE07B 0000            CON(5) =aLSTCH        LASTCHR
          0
1003 EE080 0000            CON(5) =>R            >R          ( * STR * )
          0
1004 EE085 0000            CON(5) =DUP           DUP
          0
1005 EE08A 0000            CON(5) =ONE-          1-
          0
```

```
1006 EE08F 0000          CON(5) =ZBRNH          IF          ( * STR * )
           0
1007 EE094 9100          REL(5) %IF123
           0
1008 EE099 0000          CON(5) =ONE-           1-
           0
1009 EE09E 41FD          CON(5) =aSIMFF         SIM.ASC     ( * * )
           E
1010 EE0A3 0000          CON(5) =bRNCH          ELSE
           0
1011 EE0A8 A000          REL(5) %TH123
           0
1012 EE0AD       %IF123
1013 EE0AD 0000          CON(5) =2DROP          2DROP       ( * * )
           0
1014 EE0B2       %TH123                         THEN
1015                    *
1016                    * GENERATE LAST CHARACTER (WITH HIGH BIT SET)
1017                    *
1018 EE0B2 0000          CON(5) =aLLINE         LLINE       ( * STR * )
           0
1019 EE0B7 0000          CON(5) =QUOTC          "   CON(2) #80+'"
           0
1020 EE0BC E0            CON(2) 14
1021 EE0BE E0            CON(2) 14
1022 EE0C0 0202          NIBASC \  CON(2)\
           34F4
           E482
           2392
1023 EE0D0 0232          NIBASC \ #80+'\
           8303
           B272
1024 EE0DC 0000          CON(5) =S<&            S<&         ( * STR * )
           0
1025 EE0E1 0000          CON(5) =R>             R>
           0
1026 EE0E6 0000          CON(5) =CHR$           CHR$
           0
1027 EE0EB 0000          CON(5) =S<&            S<&         ( * STR * )
           0
1028 EE0F0 AFED          CON(5) =aENDQ          ENDQUOTE
           E
1029 EE0F5 1DAD          CON(5) =aSIMLN         SIM.LINE
           E
1030 EE0FA 0000          CON(5) =SEMI           ;           ( * * )
           0
1031                    ****************************************************************
1032                    ***
1033                    ***  UPTOCFA
1034                    ***
1035                    *** ( -- )
1036                    *** ( GENERATE CODE THROUGH CODE FIELD ADDRESS )
1037                    ****************************************************************
1038 EE0FF 0000 =aUTCFA CON(5) =DOCOL          : UPTOCFA
           0
```



```
1039 EE104 0000          CON(5) =QUOTC          "   CON(5) *+5"
           0
1040 EE109 C0            CON(2) 12
1041 EE10B C0            CON(2) 12
1042 EE10D 0202          NIBASC \  \
1043 EE111 34F4          NIBASC \CON(5) *\
           E482
           5392
           02A2
1044 EE121 B253          NIBASC \+5\                        ( * STR * )
1045 EE125 62BD          CON(5) =aSTUNW         STARTUPNEW
           E
1046 EE12A 0000          CON(5) =SEMI           ;           ( * * )
           0
1047                    ****************************************************************
1048                    ***
1049                    ***  FORTHN
1050                    ***
1051                    *** ( -- [NUM] )
1052                    *** ( FORTH PRIMITIVES DEFINED IN THIS FILE )
1053                    ****************************************************************
1054 EE12F 0000 =aFORTH CON(5) =DOCOL          : FORTHN
           0
1055 EE134 3ABD          CON(5) =aFIL4T         FILE.IS.FORTH
           E
1056 EE139 0000          CON(5) =a?PAS1         ?PASS=1     ( * F * )
           0
1057 EE13E 0000          CON(5) =ZBRNH          IF          ( * * )
           0
1058 EE143 A000          REL(5) %IF124
           0
1059 EE148 0000          CON(5) =ZERO           0           ( * NUM * )
           0
1060 EE14D       %IF124                         ELSE
1061 EE14D 0000          CON(5) =SEMI           ;           ( * [NUM] * )
           0
1062                    ****************************************************************
1063                    ***
1064                    ***  WORDN
1065                    ***
1066                    *** ( -- [NUM] )
1067                    *** ( CREATE A FORTH PRIMITIVE IN THE FORTH DICTIONARY )
1068                    ****************************************************************
1069 EE152 0000 =aWORDN CON(5) =DOCOL          : WORDN
           0
1070 EE157 0000          CON(5) =a?PAS1         ?PASS=1
           0
1071 EE15C 0000          CON(5) =ZBRNH          IF
           0
1072 EE161 F000          REL(5) %IF125
           0
1073                    *
1074                    * PASS 1 PROCESSING
1075                    *
1076 EE166 F7DD          CON(5) =aCLCWD         CALCLENWORD ( * NUM * )
```

```
          E
1077 EE16B 0000         CON(5) =SEMI            ;           ( * NUM * )
          0
1078              *
1079              * PASS 2 PROCESSING
1080              *
1081 EE170         %IF125                       ELSE        ( * * )
1082 EE170 34DD           CON(5) =aENS4T        ENS.FORTH   ( * * )
          E
1083              *
1084              * GENERATE LINK FIELD
1085              *
1086 EE175 8EDD           CON(5) =aUTLFA        UPTOLFA     ( * * )
          E
1087              *
1088              * GENERATE THROUGH NAME FIELD (NON-IMMEDIATE)
1089              *
1090 EE17A 0000           CON(5) =QUOTC         "   CON(2) #80+"
          0
1091 EE17F D0            CON(2) 13
1092 EE181 D0            CON(2) 13
1093 EE183 0202          NIBASC \  CON(2)\
          34F4
          E482
          2392
1094 EE193 0232          NIBASC \ #80+\
          8303
          B2
1095 EE19D E5FD           CON(5) =aUTNFA        UPTONFA     ( * * )
          E
1096              *
1097              * GENERATE CODE FIELD
1098              *
1099 EE1A2 FF0E           CON(5) =aUTCFA        UPTOCFA
          E
1100 EE1A7 0000         CON(5) =SEMI            ;           ( * * )
          0
1101              ****************************************************************
1102              ***
1103              ***  WORDI
1104              ***
1105              *** ( -- [NUM] )
1106              *** ( CREATE A FORTH IMMEDIATE PRIMITIVE IN THE FORTH )
1107              *** ( DICTIONARY )
1108              ****************************************************************
1109 EE1AC 0000 =aWORDI CON(5) =DOCOL           : WORDI
          0
1110 EE1B1 0000         CON(5) =a?PAS1        ?PASS=1
          0
1111 EE1B6 0000         CON(5) =ZBRNH         IF          ( * * )
          0
1112 EE1BB F000         REL(5) %IF126
          0
1113              *
1114              * PASS 1 PROCESSING
```



```
1115              *
1116 EE1C0 F7DD           CON(5) =aCLCWD        CALCLENWORD ( * NUM * )
          E
1117 EE1C5 0000         CON(5) =SEMI            ;           ( * NUM * )
          0
1118              *
1119              * PASS 2 PROCESSING
1120              *
1121 EE1CA         %IF126                       ELSE        ( * * )
1122 EE1CA 34DD           CON(5) =aENS4T        ENS.FORTH   ( * * )
          E
1123              *
1124              * GENERATE LINK FIELD
1125              *
1126 EE1CF 8EDD           CON(5) =aUTLFA        UPTOLFA     ( * * )
          E
1127              *
1128              * GENERATE THROUGH NAME FIELD (IMMEDIATE)
1129              *
1130 EE1D4 0000           CON(5) =QUOTC         "   CON(2) #C0+"
          0
1131 EE1D9 D0            CON(2) 13
1132 EE1DB D0            CON(2) 13
1133 EE1DD 0202          NIBASC \  CON(2)\
          34F4
          E482
          2392
1134 EE1ED 0232          NIBASC \ #C0+\
          3403
          B2
1135 EE1F7 E5FD           CON(5) =aUTNFA        UPTONFA     ( * * )
          E
1136              *
1137              * GENERATE CODE FIELD
1138              *
1139 EE1FC FF0E           CON(5) =aUTCFA        UPTOCFA     ( * * )
          E
1140 EE201 0000         CON(5) =SEMI            ;           ( * * )
          0
1141              ****************************************************************
1142              ***
1143              ***  VALID.NAME
1144              ***
1145              *** ( STR -- STR )
1146              *** ( ENSURE NAME FIELD IS VALID )
1147              ****************************************************************
1148 EE206 0000 =aVALID CON(5) =DOCOL          : VALID.NAME
          0
1149              *
1150              * GET A STRING FROM EXPRESSION FIELD
1151              *
1152 EE20B 0000           CON(5) =aQ"STR        ".STRING    ( * STR * )
          0
1153 EE210 0000           CON(5) =aTRIM         TRIM.BLANKS ( * STR * )
          0
```

```
1154 EE215 0000         CON(5) =aENS8L        ENSURE8LONG ( * STR * )
          0
1155            *
1156            * CHECK FOR VALID SYNTAX
1157            *
1158 EE21A 0000         CON(5) =2DUP          2DUP
          0
1159 EE21F 0000         CON(5) =FSTX          FSYNTAX
          0
1160 EE224 0000         CON(5) =ZEQ           0=
          0
1161 EE229 0000         CON(5) =ZBRNH         IF          ( * STR * )
          0
1162 EE22E 4100         REL(5) %IF127
          0
1163            *
1164            * NAME ISN'T VALID HP-71 FILE NAME
1165            *
1166 EE233 0000         CON(5) =LIT           26
          0
1167 EE238 A100         CON(5) 26             'invalid filename specifier'
          0
1168 EE23D 0000         CON(5) =aSHUTD        SHUTDOWN    ( * * )
          0
1169 EE242       %IF127                       THEN        ( * STR * )
1170 EE242 0000         CON(5) =SEMI          ;           ( * STR * )
          0
1171            ****************************************************************
1172            ***
1173            ***  NAMEFIELD
1174            ***
1175            *** ( STR -- )
1176            *** ( GENERATE THE NAME FIELD OF A FILE HEADER )
1177            ****************************************************************
1178 EE247 0000 =aNAMEF CON(5) =DOCOL         : NAMEFIELD
          0
1179            *
1180            * SAVE OBJECT FILE NAME
1181            *
1182 EE24C 0000         CON(5) =2DUP          2DUP
          0
1183 EE251 0000         CON(5) =aOFILE        OFILE
          0
1184 EE256 0000         CON(5) =S!            S!
          0
1185            *
1186            * GENERATE NAME FIELD LINE
1187            *
1188 EE25B 0000         CON(5) =QUOTC         "   NIBASC '"
          0
1189 EE260 A0           CON(2) 10
1190 EE262 A0           CON(2) 10
1191 EE264 0202         NIBASC \  \
1192 EE268 E494         NIBASC \NIBASC '\
          2414
```



```
          3534
          0272
1193 EE278 0000         CON(5) =a>PAD         >PAD        ( * STR STR' * )
          0
1194 EE27D 0000         CON(5) =SWAP2         2SWAP
          0
1195 EE282 0000         CON(5) =S<&           S<&         ( * STR * )
          0
1196 EE287 AFED         CON(5) =aENDQ         ENDQUOTE
          E
1197            *
1198            * SIMULATE FILE NAME LINE
1199            *
1200 EE28C 1DAD         CON(5) =aSIMLN        SIM.LINE
          E
1201 EE291 0000         CON(5) =SEMI          ;           ( * * )
          0
1202            ****************************************************************
1203            ***
1204            ***  TIME$
1205            ***
1206            *** ( NUM -- STR )
1207            *** ( RETURN A STRING OF LENGTH TWO, BEGINNING AT 'NUM' )
1208            *** ( FROM THE BASIC STRING FUNCTION TIME$ )
1209            ****************************************************************
1210 EE296 0000 =aTIME$ CON(5) =DOCOL         : TIME$
          0
1211 EE29B 0000         CON(5) =QUOTC         " TIME$"
          0
1212 EE2A0 50           CON(2) 5
1213 EE2A2 50           CON(2) 5
1214 EE2A4 4594         NIBASC \TIME$\
          D454
          42
1215 EE2AE 0000         CON(5) =BRNCH         GOTO
          0
1216 EE2B3 D100         REL(5) %GOGET
          0
1217            ****************************************************************
1218            ***
1219            ***  DATE$
1220            ***
1221            *** ( NUM -- STR )
1222            *** ( RETURN A STRING OF LENGTH TWO, BEGINNING AT 'NUM' )
1223            *** ( FROM THE BASIC STRING FUNCTION DATE$ )
1224            ****************************************************************
1225 EE2B8 0000 =aDATE$ CON(5) =DOCOL         : DATE$
          0
1226 EE2BD 0000         CON(5) =QUOTC         " DATE$"
          0
1227 EE2C2 50           CON(2) 5
1228 EE2C4 50           CON(2) 5
1229 EE2C6 4414         NIBASC \DATE$\
          4554
          42
```

```
1230 EE2D0 0000  %GOGET CON(5) =aBAS$          BASIC$       ( * NUM STR * )
           0
1231 EE2D5 0000          CON(5) =ROT           ROT          ( * STR NUM * )
           0
1232 EE2DA 0000          CON(5) =END$          END$
           0
1233 EE2DF 0000          CON(5) =DROP          DROP
           0
1234 EE2E4 0000          CON(5) =TWO           2
           0
1235 EE2E9 0000          CON(5) =SEMI          ;            ( * STR * )
           0
1236             ****************************************************************
1237             ***
1238             ***  TIME/DATE
1239             ***
1240             *** ( -- )
1241             *** ( GENERATE THE TIME AND DATE FIELD OF A FILE HEADER )
1242             ****************************************************************
1243 EE2EE 0000 =aTI/DA CON(5) =DOCOL          : TIME/DATE
           0
1244             *
1245             * MINUTES FIELD
1246             *
1247 EE2F3 0000          CON(5) =aLLINE        LLINE        ( * STR * )
           0
1248 EE2F8 0000          CON(5) =QUOTC         "   CON(2) #"
           0
1249 EE2FD A0            CON(2) 10
1250 EE2FF A0            CON(2) 10
1251 EE301 0202          NIBASC \  \
1252 EE305 34F4          NIBASC \CON(2) #\
           E482
           2392
           0232
1253 EE315 0000          CON(5) =S<&           S<&          ( * STR * )
           0
1254 EE31A 0000          CON(5) =LIT           4
           0
1255 EE31F 4000          CON(5) 4
           0
1256 EE324 692E          CON(5) =aTIME$        TIME$
           E
1257 EE329 0000          CON(5) =S<&           S<&
           0
1258 EE32E 1DAD          CON(5) =aSIMLN        SIM.LINE     ( * * )
           E
1259             *
1260             * HOUR FIELD
1261             *
1262 EE333 0000          CON(5) =aLLINE        LLINE        ( * STR * )
           0
1263 EE338 0000          CON(5) =QUOTC         "   CON(2) #"
           0
1264 EE33D A0            CON(2) 10
```



```
1265 EE33F A0            CON(2) 10
1266 EE341 0202          NIBASC \  \
1267 EE345 34F4          NIBASC \CON(2) #\
           E482
           2392
           0232
1268 EE355 0000          CON(5) =S<&           S<&          ( * STR * )
           0
1269 EE35A 0000          CON(5) =ONE           1
           0
1270 EE35F 692E          CON(5) =aTIME$        TIME$
           E
1271 EE364 0000          CON(5) =S<&           S<&
           0
1272 EE369 1DAD          CON(5) =aSIMLN        SIM.LINE     ( * * )
           E
1273             *
1274             * DAY FIELD
1275             *
1276 EE36E 0000          CON(5) =aLLINE        LLINE        ( * STR * )
           0
1277 EE373 0000          CON(5) =QUOTC         "   CON(2) #"
           0
1278 EE378 A0            CON(2) 10
1279 EE37A A0            CON(2) 10
1280 EE37C 0202          NIBASC \  \
1281 EE380 34F4          NIBASC \CON(2) #\
           E482
           2392
           0232
1282 EE390 0000          CON(5) =S<&           S<&          ( * STR * )
           0
1283 EE395 0000          CON(5) =LIT           7
           0
1284 EE39A 7000          CON(5) 7
           0
1285 EE39F 8B2E          CON(5) =aDATE$        DATE$
           E
1286 EE3A4 0000          CON(5) =S<&           S<&
           0
1287 EE3A9 1DAD          CON(5) =aSIMLN        SIM.LINE     ( * * )
           E
1288             *
1289             * MONTH FIELD
1290             *
1291 EE3AE 0000          CON(5) =aLLINE        LLINE        ( * STR * )
           0
1292 EE3B3 0000          CON(5) =QUOTC         "   CON(2) #"
           0
1293 EE3B8 A0            CON(2) 10
1294 EE3BA A0            CON(2) 10
1295 EE3BC 0202          NIBASC \  \
1296 EE3C0 34F4          NIBASC \CON(2) #\
           E482
           2392
```

```
          0232
1297 EE3D0 0000         CON(5) =S<&             S<&         ( * STR * )
          0
1298 EE3D5 0000         CON(5) =LIT             4
          0
1299 EE3DA 4000         CON(5) 4
          0
1300 EE3DF 8B2E         CON(5) =aDATE$          DATE$
          E
1301 EE3E4 0000         CON(5) =S<&             S<&
          0
1302 EE3E9 1DAD         CON(5) =aSIMLN          SIM.LINE    ( * * )
          E
1303            *
1304            * YEAR FIELD
1305            *
1306 EE3EE 0000         CON(5) =aLLINE          LLINE       ( * STR * )
          0
1307 EE3F3 0000         CON(5) =QUOTC           "   CON(2) #"
          6
1308 EE3F8 A0           CON(2) 10
1309 EE3FA A0           CON(2) 10
1310 EE3FC 0202         NIBASC \  \
1311 EE400 34F4         NIBASC \CON(2) #\
          E482
          2392
          0232
1312 EE410 0000         CON(5) =S<&             S<&         ( * STR * )
          0
1313 EE415 0000         CON(5) =ONE             1
          0
1314 EE41A 8B2E         CON(5) =aDATE$          DATE$
          E
1315 EE41F 0000         CON(5) =S<&             S<&
          0
1316 EE424 1DAD         CON(5) =aSIMLN          SIM.LINE
          E
1317 EE429 0000         CON(5) =SEMI            ;           ( * * )
          0
1318            ***************************************************************
1319            ***
1320            ***  LEX
1321            ***
1322            *** ( -- [NUM] )
1323            *** ( CREATE A LEX FILE HEADER )
1324            ***************************************************************
1325 EE42E 0000 =aLEX   CON(5) =DOCOL          : LEX
          0
1326            *
1327            * CHECK FOR FILETYPE & MULTIPLE LEX PSEUDO-OPS
1328            *
1329 EE433 A8CD         CON(5) =aFILLX          FILE.IS.LEX ( * * )
          E
1330 EE438 0000         CON(5) =a?PAS1          ?PASS=1
          0
```



```
1331 EE43D 0000         CON(5) =ZBRNH           IF          ( * * )
          0
1332 EE442 A500         REL(5) %IF128
          0
1333            *
1334            * PASS 1
1335            *
1336            *
1337            * ENSURE FILE NAME VALID & REMOVE OLD LEX FILE
1338            *
1339 EE447 0000         CON(5) =aEXP.F          EXPRESSION.FIELD
          0
1340 EE44C 602E         CON(5) =aVALID          VALID.NAME  ( * STR * )
          E
1341 EE451 0000         CON(5) =LIT             lex file type
          0
1342 EE456 802E         CON(5) #E208
          0
1343 EE45B 0000         CON(5) =aPURGE          PURGE
          0
1344 EE460 0000         CON(5) =DROP            DROP        ( * * )
          0
1345            *
1346            * RESET PROGRESS VARIABLE BECAUSE PURGING A LEX FILE WIPES OUT TH
1347            * DISPLAY BUFFER
1348            *
1349 EE465 0000         CON(5) =ZERO            0
          0
1350 EE46A 0000         CON(5) =aLINEC          LINECOUNT
          0
1351 EE46F 0000         CON(5) =STORE           !
          0
1352            *
1353            * UPDATE POINTERS TO REFLECT MEMORY MOVE
1354            *
1355 EE474 0000         CON(5) =aMEMOV          MEMORY MOVED
          0
1356 EE479 0000         CON(5) =aOBJFL          OBJ.FILE
          0
1357 EE47E 0000         CON(5) =AT              @
          0
1358 EE483 0000         CON(5) =aLC             LC
          0
1359 EE488 0000         CON(5) =STORE           !
          0
1360            *
1361            * RETURN LENGTH
1362            *
1363 EE48D 0000         CON(5) =LIT             37          ( * NUM * )
          0
1364 EE492 5200         CON(5) 37
          0
1365 EE497 0000         CON(5) =SEMI            ;           ( * NUM * )
          0
1366 EE49C       %IF128                         ELSE        ( * * )
```

```
1367            *
1368            * PASS 2
1369            *
1370            *
1371            * CHECK VALID NAME & OUTPUT NAME FIELD
1372            *
1373 EE49C 0000         CON(5) =aOLDEX          OLD.EXPR
          0
1374 EE4A1 602E         CON(5) =aVALID          VALID.NAME  ( * STR * )
          E
1375 EE4A6 742E         CON(5) =aNAMEF          NAMEFIELD   ( * * )
          E
1376            *
1377            * OUTPUT FILE TYPE LINE
1378            *
1379 EE4AB 0000         CON(5) =QUOTC           "   NIBHEX 802E"
          0
1380 EE4B0 D0           CON(2) 13
1381 EE4B2 D0           CON(2) 13
1382 EE4B4 0202         NIBASC \  \
1383 EE4B8 E494         NIBASC \NIBHEX 8\
          2484
          5485
          0283
1384 EE4C8 0323         NIBASC \02E\
          54
1385 EE4CE 1DAD         CON(5) =aSIMLN          SIM.LINE    ( * * )
          E
1386            *
1387            * OUTPUT FLAGS LINE
1388            *
1389 EE4D3 0000         CON(5) =QUOTC           "   CON(2) 0"
          0
1390 EE4D8 A0           CON(2) 10
1391 EE4DA A0           CON(2) 10
1392 EE4DC 0202         NIBASC \  \
1393 EE4E0 34F4         NIBASC \CON(2) 0\
          E482
          2392
          0203
1394 EE4F0 1DAD         CON(5) =aSIMLN          SIM.LINE    ( * * )
          E
1395            *
1396            * OUTPUT TIME AND DATE LINES
1397            *
1398 EE4F5 EE2E         CON(5) =aTI/DA          TIME/DATE   ( * * )
          E
1399            *
1400            * OUTPUT FILE CHAIN LINK LINE
1401            *
1402 EE4FA 0000         CON(5) =QUOTC           "   REL(5) FiLeNd"
          0
1403 EE4FF F0           CON(2) 15
1404 EE501 F0           CON(2) 15
1405 EE503 0202         NIBASC \  \
```



```
1406 EE507 2554         NIBASC \REL(5) F\
          C482
          5392
          0264
1407 EE517 96C4         NIBASC \iLeNd\
          56E4
          46
1408 EE521 62BD         CON(5) =aSTUNW          STARTUPNEW  ( * * )
          E
1409 EE526 0000         CON(5) =SEMI            ;           ( * * )
          0
1410            ****************************************************************
1411            ***
1412            ***  ID
1413            ***
1414            *** ( -- [NUM] )
1415            *** ( APPEND LEX FILE ID TO HEADER )
1416            ****************************************************************
1417 EE52B 0000 =aIDN   CON(5) =DOCOL           : ID
          0
1418 EE530 0000         CON(5) =a?PAS1          ?PASS=1
          0
1419 EE535 0000         CON(5) =ZBRNH           IF
          0
1420 EE53A 4100         REL(5) %IF129
          0
1421 EE53F 0000         CON(5) =LIT             16          ( * NUM * )
          0
1422            *
1423            * PASS 1 PROCESSING
1424            *
1425 EE544 0100         CON(5) 16
          0
1426 EE549 0000         CON(5) =SEMI            ;           ( * NUM * )
          0
1427            *
1428            * PASS 2 PROCESSING
1429            *
1430 EE54E       %IF129                         ELSE        ( * * )
1431 EE54E 75DD         CON(5) =aENSLX          ENSURE.LEX  ( * * )
          E
1432 EE553 FEAD         CON(5) =aWRUOL          WRAPUPOLD   ( * * )
          E
1433            *
1434            * GENERATE ID FIELD
1435            *
1436 EE558 0000         CON(5) =QUOTC           "   CON(2) "
          0
1437 EE55D 90           CON(2) 9
1438 EE55F 90           CON(2) 9
1439 EE561 0202         NIBASC \  \
1440 EE565 34F4         NIBASC \CON(2) \
          E482
          2392
          02
```

```
1441 EE573 0000         CON(5) =a>PAD          >PAD         ( * STR * )
           0
1442 EE578 0000         CON(5) =aOLDEX         OLDEXPR
           0
1443 EE57D 0000         CON(5) =S<&            S<&          ( * STR * )
           0
1444 EE582 1DAD         CON(5) =aSIMLN         SIM.LINE     ( * * )
           E
1445            *
1446            * GENERATE LOW TOKEN FIELD
1447            *
1448 EE587 0000         CON(5) =QUOTC          "   CON(2) 0"
           0
1449 EE58C A0           CON(2) 10
1450 EE58E A0           CON(2) 10
1451 EE590 0202         NIBASC \  \
1452 EE594 34F4         NIBASC \CON(2) 0\              ( * STR * )
           E482
           2392
           0203
1453 EE5A4 0000         CON(5) =LIT            16
           0
1454 EE5A9 0100         CON(5) 16
           0
1455 EE5AE 0000         CON(5) =aTESTF         TEST.FLAG    ( * STR F * )
           0
1456 EE5B3 0000         CON(5) =ZBRNH          IF           ( * STR * )
           0
1457 EE5B8 E100         REL(5) %IF152
           0
1458 EE5BD 0000         CON(5) =a>PAD          >PAD
           0
1459 EE5C2 0000         CON(5) =aLOTOK         LOTOK
           0
1460 EE5C7 0000         CON(5) =AT             @
           0
1461 EE5CC 0000         CON(5) =aNUM#3         #3           ( * STR STR' * )
           0
1462 EE5D1 0000         CON(5) =S<&            S<&
           0
1463 EE5D6      %IF152                         THEN         ( * STR * )
1464 EE5D6 1DAD         CON(5) =aSIMLN         SIM.LINE     ( * * )
           E
1465            *
1466            * GENERATE HIGH TOKEN FIELD
1467            *
1468 EE5DB 0000         CON(5) =QUOTC          "   CON(2) 0"
           0
1469 EE5E0 A0           CON(2) 10
1470 EE5E2 A0           CON(2) 10
1471 EE5E4 0202         NIBASC \  \
1472 EE5E8 34F4         NIBASC \CON(2) 0\
           E482
           2392
           0203
```



```
1473 EE5F8 0000         CON(5) =LIT            16
           0
1474 EE5FD 0100         CON(5) 16
           0
1475 EE602 0000         CON(5) =aTESTF         TEST.FLAG    ( * STR F * )
           0
1476 EE607 0000         CON(5) =ZBRNH          IF           ( * STR * )
           0
1477 EE60C E100         REL(5) %IF153
           0
1478 EE611 0000         CON(5) =a>PAD          >PAD
           0
1479 EE616 0000         CON(5) =aHITOK         HITOK
           0
1480 EE61B 0000         CON(5) =AT             @
           0
1481 EE620 0000         CON(5) =aNUM#3         #3
           0
1482 EE625 0000         CON(5) =S<&            S<&
           0
1483 EE62A      %IF153                         THEN         ( * STR * )
1484 EE62A 1DAD         CON(5) =aSIMLN         SIM.LINE     ( * * )
           E
1485            *
1486            * NULL LEX TABLE LINK
1487            *
1488 EE62F 0000         CON(5) =QUOTC          "   CON(5) 0"
           0
1489 EE634 A0           CON(2) 10
1490 EE636 A0           CON(2) 10
1491 EE638 0202         NIBASC \  \
1492 EE63C 34F4         NIBASC \CON(5) 0\
           E482
           5392
           0203
1493 EE64C 1DAD         CON(5) =aSIMLN         SIM.LINE     ( * * )
           E
1494            *
1495            * NO SPEED TABLE
1496            *
1497 EE651 0000         CON(5) =QUOTC          "   NIBHEX F"
           0
1498 EE656 A0           CON(2) 10
1499 EE658 A0           CON(2) 10
1500 EE65A 0202         NIBASC \  \
1501 EE65E E494         NIBASC \NIBHEX F\
           2484
           5485
           0264
1502 EE66E 1DAD         CON(5) =aSIMLN         SIM.LINE     ( * * )
           E
1503            *
1504            * OFFSET TO TEXT TABLE
1505            *
1506 EE673 0000         CON(5) =QUOTC          " REL(4) 1+TxTbSt"
```

```
          0
1507 EE678 11          CON(2) 17
1508 EE67A 11          CON(2) 17
1509 EE67C 0202        NIBASC \  \
1510 EE680 2554        NIBASC \REL(4) 1\
           C482
           4392
           0213
1511 EE690 B245        NIBASC \+TxTbSt\
           8745
           2635
           47
1512 EE69E 62BD        CON(5) =aSTUNW        STARTUPNEW  ( * * )
           E
1513 EE6A3 0000        CON(5) =SEMI          ;           ( * * )
           0
1514           ****************************************************************
1515           ***
1516           ***  MSG
1517           ***
1518           *** ( -- )
1519           *** ( APPEND MESSAGE TABLE OFFSET TO HEADER )
1520           ****************************************************************
1521 EE6A8 0000 =aMSG   CON(5) =DOCOL         : MSG
           0
1522 EE6AD 75DD        CON(5) =aENSLX        ENSURE.LEX  ( * * )
           E
1523 EE6B2 FEAD        CON(5) =aWRUOL        WRAPUPOLD   ( * * )
           E
1524 EE6B7 0000        CON(5) =aEXPR         EXPR
           0
1525 EE6BC 0000        CON(5) =ZBRNH         IF          ( * * )
           0
1526 EE6C1 9300        REL(5) %IF130
           0
1527           *
1528           * USER SPECIFIED A VALID LABEL
1529           *
1530 EE6C6 0000        CON(5) =QUOTC         "   REL(4) "
           0
1531 EE6CB 90          CON(2) 9
1532 EE6CD 90          CON(2) 9
1533 EE6CF 0202        NIBASC \  \
1534 EE6D3 2554        NIBASC \REL(4) \
           C482
           4392
           02
1535 EE6E1 0000        CON(5) =a>PAD         >PAD        ( * STR * )
           0
1536 EE6E6 0000        CON(5) =aOLDEX        OLDEXPR
           0
1537 EE6EB 0000        CON(5) =S<&           S<&         ( * STR * )
           0
1538 EE6F0 0000        CON(5) =BRNCH
           0
```



```
1539 EE6F5 2200        REL(5) %TH130
           0
1540 EE6FA       %IF130                      ELSE        ( * * )
1541           *
1542           * USER SPECIFIED AN INVALID LABEL, OR ZERO
1543           *
1544 EE6FA 0000        CON(5) =QUOTC         "   CON(4) 0"
           0
1545 EE6FF A0          CON(2) 10
1546 EE701 A0          CON(2) 10
1547 EE703 0202        NIBASC \  \
1548 EE707 34F4        NIBASC \CON(4) 0\     ( * STR * )
           E482
           4392
           0203
1549           *
1550           * SIMULATE THIS LINE
1551           *
1552 EE717       %TH130                      THEN
1553 EE717 62BD        CON(5) =aSTUNW        STARTUPNEW  ( * * )
           E
1554 EE71C 0000        CON(5) =SEMI          ;           ( * * )
           0
1555           ****************************************************************
1556           ***
1557           ***  POLL
1558           ***
1559           *** ( -- )
1560           *** ( APPEND POLL TABLE OFFSET TO HEADER )
1561           ****************************************************************
1562 EE721 0000 =aPOLL  CON(5) =DOCOL         : POLL
           0
1563 EE726 75DD        CON(5) =aENSLX        ENSURE.LEX  ( * * )
           E
1564 EE72B FEAD        CON(5) =aWRUOL        WRAPUPOLD   ( * * )
           E
1565 EE730 0000        CON(5) =aEXPR         EXPR
           0
1566 EE735 0000        CON(5) =ZBRNH         IF          ( * * )
           0
1567 EE73A 9300        REL(5) %IF131
           0
1568           *
1569           * USER SPECIFIED A VALID LABEL
1570           *
1571 EE73F 0000        CON(5) =QUOTC         "   REL(5) "
           0
1572 EE744 90          CON(2) 9
1573 EE746 90          CON(2) 9
1574 EE748 0202        NIBASC \  \
1575 EE74C 2554        NIBASC \REL(5) \
           C482
           5392
           02
1576 EE75A 0000        CON(5) =a>PAD         >PAD        ( * STR * )
```

```
          0
1577 EE75F 0000         CON(5) =aOLDEX        OLDEXPR
          0
1578 EE764 0000         CON(5) =S<&           S<&         ( * STR * )
          0
1579 EE769 0000         CON(5) =BRNCH
          0
1580 EE76E 2200         REL(5) %TH131
          0
1581                *
1582                * USER SPECIFIED AN INVALID LABEL OR ZERO
1583                *
1584 EE773              %IF131                      ELSE        ( * * )
1585 EE773 0000         CON(5) =QUOTC         "   CON(5) 0"
          0
1586 EE778 A0           CON(2) 10
1587 EE77A A0           CON(2) 10
1588 EE77C 0202         NIBASC \  \
1589 EE780 34F4         NIBASC \CON(5) 0\             ( * STR * )
          E482
          5392
          0203
1590                *
1591                * SIMULATE THIS LINE
1592                *
1593 EE790              %TH131                      THEN        ( * STR * )
1594 EE790 62BD         CON(5) =aSTUNW        STAKTUPNEW  ( * * )
          E
1595 EE795 0000         CON(5) =SEMI          ;           ( * * )
          0
1596                ****************************************************************
1597                ***
1598                ***  ENTRY
1599                ***
1600                *** ( -- )
1601                *** ( ENTRY POINT INTO HEADER )
1602                ****************************************************************
1603 EE79A 0000 =aENTRY CON(5) =DOCOL          : ENTRY
          0
1604 EE79F 75DD         CON(5) =aENSLX        ENSURE.LEX  ( * * )
          E
1605 EE7A4 FEAD         CON(5) =aWRUOL        WRAPUPOLD   ( * * )
          E
1606                *
1607                * HAVE WE GENERATED THE MAIN TABLE YET?
1608                *
1609 EE7A9 0000         CON(5) =aTENT         TENT
          0
1610 EE7AE 0000         CON(5) =AT            @
          0
1611 EE7B3 0000         CON(5) =ZEQ           0=
          0
1612 EE7B8 0000         CON(5) =ZBRNH         IF          ( * * )
          0
1613 EE7BD 5500         REL(5) %IF132
```



```
          0
1614                *
1615                * NO.  LET'S DO SO NOW.
1616                *
1617 EE7C2 0000         CON(5) =QUOTC         "   * * * M A I N   T A B L E * * *
          0
1618 EE7C7 12           CON(2) 33
1619 EE7C9 12           CON(2) 33
1620 EE7CB 0202         NIBASC \  \
1621 EE7CF A202         NIBASC \* * * M \
          A202
          A202
          D402
1622 EE7DF 1402         NIBASC \A I N   \
          9402
          E402
          0202
1623 EE7EF 4502         NIBASC \T A B L \
          1402
          2402
          C402
1624 EE7FF 5402         NIBASC \E * * *\              ( * STR * )
          A202
          A202
          A2
1625 EE80D 1DAD         CON(5) =aSIMLN        SIM.LINE
          E
1626                *
1627                * ONE MORE ENTRY TO THE MAIN TABLE
1628                *
1629 EE812              %IF132                      THEN        ( * * )
1630 EE812 0000         CON(5) =ONE           1
          0
1631 EE817 0000         CON(5) =aTENT         TENT
          0
1632 EE81C 0000         CON(5) =PSTOR         +!
          0
1633                *
1634                * GENERATE THREE NIBBLE OFFSET TO CORRECT ENTRY IN TEXT TABLE
1635                *
1636 EE821 0000         CON(5) =QUOTC         "   CON(3) (TxEn"
          0
1637 EE826 E0           CON(2) 14
1638 EE828 E0           CON(2) 14
1639 EE82A 0202         NIBASC \  \
1640 EE82E 34F4         NIBASC \CON(3) (\
          E482
          3392
          0282
1641 EE83E 4587         NIBASC \IxEn\
          54E6
1642 EE846 0000         CON(5) =a>PAD         >PAD        ( * STR * )
          0
1643 EE84B 0000         CON(5) =aTENT         TENT
          0
```

```
1644 EE850 0000          CON(5) =AT            @
           0
1645 EE855 0000          CON(5) =aNUM#2        #2
           0
1646 EE85A 0000          CON(5) =S<&           S<&         ( * STR * )
           0
1647 EE85F 0000          CON(5) =QUOTC         " )-(TxTbSt)"
           0
1648 EE864 A0            CON(2) 10
1649 EE866 A0            CON(2) 10
1650 EE868 92D2          NIBASC \)-(TxTbS\
           8245
           8745
           2635
1651 EE878 4792          NIBASC \t)\
1652 EE87C 0000          CON(5) =S<&           S<&         ( * STR * )
           0
1653 EE881 1DAD          CON(5) =aSIMLN        SIM.LINE    ( * * )
           E
1654               *
1655               * GENERATE FIVE NIBBLE OFFSET TO ROUTINE TO HANDLE THIS WORD
1656               *
1657 EE886 0000          CON(5) =QUOTC         "
           0
1658 EE88B 90            CON(2) 9
1659 EE88D 90            CON(2) 9
1660 EE88F 0202          NIBASC \  \
1661 EE893 2554          NIBASC \REL(5) \
           C482
           5392
           02
1662 EE8A1 0000          CON(5) =a>PAD         >PAD        ( * STR * )
           0
1663 EE8A6 0000          CON(5) =aOLDEX        OLDEXPR
           0
1664 EE8AB 0000          CON(5) =S<&           S<&         ( * STR * )
           0
1665 EE8B0 62BD          CON(5) =aSTUNW        STARTUPNEW
           E
1666 EE8B5 0000          CON(5) =SEMI          ;           ( * * )
           0
1667               ****************************************************************
1668               ***
1669               ***  CHAR
1670               ***
1671               *** ( -- )
1672               *** ( APPEND CHARACTERISTIC NIBBLE TO HEADER )
1673               ****************************************************************
1674 EE8BA 0000 =aCHAR  CON(5) =DOCOL         : CHAR
           0
1675 EE8BF 75DD          CON(5) =aENSLX        ENSURE.LEX  ( * * )
           E
1676 EE8C4 FEAD          CON(5) =aWRUOL        WRAPUPOLD   ( * * )
           E
1677               *
```



```
1678               * GENERATE ONE NIBBLE CONSTANT WITH CHARACTERISTIC NIBBLE
1679               *
1680 EE8C9 0000          CON(5) =QUOTC         "   CON(1) "
           0
1681 EE8CE 90            CON(2) 9
1682 EE8D0 90            CON(2) 9
1683 EE8D2 0202          NIBASC \  \
1684 EE8D6 34F4          NIBASC \CON(1) \
           E482
           1392
           02
1685 EE8E4 0000          CON(5) =a>PAD         >PAD        ( * STR * )
           0
1686 EE8E9 0000          CON(5) =aOLDEX        OLDEXPR
           0
1687 EE8EE 0000          CON(5) =S<&           S<&         ( * STR * )
           0
1688               *
1689               * SIMULATE THIS LINE
1690               *
1691 EE8F3 62BD          CON(5) =aSTUNW        STARTUPNEW
           E
1692 EE8F8 0000          CON(5) =SEMI          ;           ( * * )
           0
1693               ****************************************************************
1694               ***
1695               ***  GENTxEnxx
1696               ***
1697               *** ( -- STR )
1698               *** ( GENERATE A STRING "TxEn01" OR WHATEVER IS IN KENT )
1699               ****************************************************************
1700 EE8FD 0000 =aGENTx CON(5) =DOCOL         : GENTxEnxx
           0
1701               *
1702               * PUT TxEn ONTO PAD
1703               *
1704 EE902 0000          CON(5) =QUOTC         " TxEn"
           0
1705 EE907 40            CON(2) 4
1706 EE909 40            CON(2) 4
1707 EE90B 4587          NIBASC \TxEn\
           54E6
1708 EE913 0000          CON(5) =a>PAD         >PAD        ( * STR * )
           0
1709               *
1710               * APPEND TWO DIGIT STRING ONTO PAD
1711               *
1712 EE918 0000          CON(5) =aKENT         KENT
           0
1713 EE91D 0000          CON(5) =AT            @
           0
1714 EE922 0000          CON(5) =aNUM#2        #2
           0
1715 EE927 0000          CON(5) =S<&           S<&
           0
```

```
1716 EE92C 0000         CON(5) =SEMI          ;           ( * STR * )
           0
1717            ****************************************************************
1718            ***
1719            ***  TXTBST
1720            ***
1721            *** ( -- STR )
1722            *** ( GENERATE THE STRING 'TxTbSt' )
1723            ****************************************************************
1724 EE931 0000 =TxTbSt CON(5) =DOCOL         : TXTBST
           0
1725 EE936 0000         CON(5) =QUOTC         " TxTbSt"
           0
1726 EE93B 60           CON(2) 6
1727 EE93D 60           CON(2) 6
1728 EE93F 4587         NIBASC \TxTbSt\
           4526
           3547
1729 EE94B 0000         CON(5) =SEMI          ;           ( * STR * )
           0
1730            ****************************************************************
1731            ***
1732            ***  KEYP
1733            ***
1734            *** ( -- [NUM] )
1735            *** ( APPEND KEYWORD TEXT TO HEADER )
1736            ****************************************************************
1737 EE950 0000 =aKEYP  CON(5) =DOCOL         : KEYP
           0
1738 EE955 0000         CON(5) =a?PAS1        ?PASS=1
           0
1739 EE95A 0000         CON(5) =ZBRNH         IF          ( * * )
           0
1740 EE95F D700         REL(5) %IF133
           0
1741            *
1742            * PASS 1 PROCESSING
1743            *
1744 EE964 0000         CON(5) =aKENT         KENT
           0
1745 EE969 0000         CON(5) =AT            @
           0
1746 EE96E 0000         CON(5) =ZEQ           0=
           0
1747 EE973 0000         CON(5) =ZBRNH         IF          ( * * )
           0
1748 EE978 9100         REL(5) %IF134
           0
1749            *
1750            * FIRST KEY PSEUDO-OP, PUT TEXT TABLE START ON SYMBOL TABLE
1751            *
1752 EE97D 0000         CON(5) =aLC           LC
           0
1753 EE982 0000         CON(5) =AT            @
           0
```



```
1754 EE987 139E         CON(5) =TxTbSt        TXTBST      ( * NUM STR * )
           E
1755 EE98C 0000         CON(5) =a>SYT         >SYT
           0
1756            *
1757            * ONE MORE KEY PSUEDO-OP
1758            *
1759 EE991       %IF134                       THEN        ( * * )
1760 EE991 0000         CON(5) =ONE           1
           0
1761 EE996 0000         CON(5) =aKENT         KENT
           0
1762 EE99B 0000         CON(5) =STOR          +!          ( * * )
           0
1763            *
1764            * GENERATE TXENXX LABEL IN SYMBOL TABLE
1765            *
1766 EE9A0 0000         CON(5) =aLC           LC
           0
1767 EE9A5 0000         CON(5) =AT            @
           0
1768 EE9AA DF8E         CON(5) =aGENTx        GENTxEnxx
           E
1769 EE9AF 0000         CON(5) =a>SYT         >SYT        ( * * )
           0
1770            *
1771            * CALCULATE LENGTH OF KEY PSEUDO-OP
1772            *
1773 EE9B4 0000         CON(5) =aEXP.F        EXPRESSION.FIELD
           0
1774 EE9B9 0000         CON(5) =aQ"STR        QUOTED.STRING
           0
1775 EE9BE 0000         CON(5) =aTRIM         TRIM.BLANKS
           0
1776 EE9C3 0000         CON(5) =SWAP          SWAP
           0
1777 EE9C8 0000         CON(5) =DROP          DROP        ( * NUM * )
           0
1778 EE9CD 0000         CON(5) =TWO*          2*          ( * NUM * )
           0
1779 EE9D2 0000         CON(5) =ONE+          1+          ( * NUM * )
           0
1780 EE9D7 0000         CON(5) =SEMI          ;           ( * NUM * )
           0
1781            *
1782            * PASS 2 PROCESSING
1783            *
1784 EE9DC       %IF133                       ELSE        ( * * )
1785 EE9DC 75DD         CON(5) =aENSLX        ENSURE.LEX  ( * * )
           E
1786 EE9E1 FEAD         CON(5) =aWRUOL        WRAPUPOLD   ( * * )
           E
1787 EE9E6 0000         CON(5) =aKENT         KENT
           0
1788 EE9EB 0000         CON(5) =AT            @
```

```
          0
1789 EE9F0 0000        CON(5) =ZEQ          0=
          0
1790 EE9F5 0000        CON(5) =ZBRNH        IF          ( * * )
          0
1791 EE9FA F500        REL(5) %IF135
          0
1792           *
1793           * FIRST KEY PSEUDO-OP (PASS2), GENERATE START OF TEXT TABLE
1794           *
1795 EE9FF 0000        CON(5) =QUOTC        "   * * * T E X T   T A B L E * * *
          0
1796 EEA04 12          CON(2) 33
1797 EEA06 12          CON(2) 33
1798 EEA08 0202        NIBASC \  \
1799 EEA0C A202        NIBASC \* * * T \
          A202
          A202
          4502
1800 EEA1C 5402        NIBASC \E X T   \
          8502
          4502
          0202
1801 EEA2C 4502        NIBASC \T A B L \
          1402
          2402
          C402
1802 EEA3C 5402        NIBASC \E * * *\
          A202
          A202
          A2
1803 EEA4A 1DAD        CON(5) =aSIMLN       SIM.LINE    ( * * )
          E
1804 EEA4F 139E        CON(5) =TXTBST       TXTBST      ( * STR * )
          E
1805 EEA54 1DAD        CON(5) =aSIMLN       SIM.LINE    ( * * )
          E
1806           *
1807           * ONE MORE KEY PSEUDO-OP
1808           *
1809 EEA59      %IF135                       THEN        ( * * )
1810 EEA59 0000        CON(5) =ONE          1
          0
1811 EEA5E 0000        CON(5) =aKENT        KENT
          0
1812 EEA63 0000        CON(5) =PSTOR        +!          ( * * )
          0
1813           *
1814           * GENERATE LINE FOR LABEL (LABEL IN DURING 1ST PASS)
1815           *
1816 EEA68 DF8E        CON(5) =aGENTx       GENTxEnxx   ( * STR * )
          E
1817 EEA6D 1DAD        CON(5) =aSIMLN       SIM.LINE    ( * * )
          E
1818           *
```



```
1819           * GENERATE ONE NIBBLE CONSTANT OF LENGTH - 1
1820           *
1821 EEA72 0000        CON(5) =QUOTC        "   CON(1) "
          0
1822 EEA77 90          CON(2) 9
1823 EEA79 90          CON(2) 9
1824 EEA7B 0202        NIBASC \  \
1825 EEA7F 34F4        NIBASC \CON(1) \
          E482
          1392
          02
1826 EEA8D 0000        CON(5) =a>PAD        >PAD        ( * STR * )
          0
1827 EEA92 0000        CON(5) =aOLDEX       OLDEXPR
          0
1828 EEA97 0000        CON(5) =aQ"STR       QUOTED.STRING
          0
1829 EEA9C 0000        CON(5) =aTRIM        TRIM.BLANKS
          0
1830           *
1831           * RESAVE THE OLD EXPRESSION WITHOUT QUOTES OR BLANKS
1832           *
1833 EEAA1 0000        CON(5) =2DUP         2DUP
          0
1834 EEAA6 0000        CON(5) =aOLDEX       OLDEXPR
          0
1835 EEAAB 0000        CON(5) =S!           S!          ( * STR * )
          0
1836 EEAB0 0000        CON(5) =SWAP         SWAP
          0
1837 EEAB5 0000        CON(5) =DROP         DROP        ( * STR LEN * )
          0
1838 EEABA 0000        CON(5) =TWO*         2*
          0
1839 EEABF 0000        CON(5) =ONE-         1-
          0
1840 EEAC4 0000        CON(5) =aNUM#2       #2
          0
1841 EEAC9 0000        CON(5) =S<&          S<&         ( * STR * )
          0
1842 EEACE 1DAD        CON(5) =aSIMLN       SIM.LINE    ( * * )
          E
1843           *
1844           * GENERATE ASCII FOR KEY PSEUDO-OP
1845           *
1846 EEAD3 0000        CON(5) =QUOTC        "   NIBASC '"
          0
1847 EEAD8 A0          CON(2) 10
1848 EEADA A0          CON(2) 10
1849 EEADC 0202        NIBASC \  NIBASC\
          E494
          2414
          3534
1850 EEAEC 0272        NIBASC \ '\
1851 EEAF0 0000        CON(5) =a>PAD        >PAD        ( * STR * )
```

```
          0
1852 EEAF5 0000         CON(5) =aOLDEX        OLDEXPR
          0
1853 EEAFA 0000         CON(5) =S<&           S<&         ( * STR * )
          0
1854 EEAFF AFED         CON(5) =aENDQ         END.QUOTE
          E
1855            *
1856            * SIMULATE THIS LINE
1857            *
1858 EEB04 62BD         CON(5) =aSTUNW        STARTUPNEW  ( * * )
          E
1859 EEB09 0000         CON(5) =SEMI          ;           ( * * )
          0
1860            ****************************************************************
1861            ***
1862            ***  TOKENP
1863            ***
1864            *** ( -- [NUM] )
1865            *** ( APPEND TOKEN TO HEADER )
1866            ****************************************************************
1867 EEB0E 0000 =aTOKNP CON(5) =DOCOL          : TOKENP
          0
1868 EEB13 0000         CON(5) =a?PAS1        ?PASS=1
          0
1869 EEB18 0000         CON(5) =ZBRNH         IF          ( * * )
          0
1870 EEB1D A500         REL(5) %IF136
          0
1871            *
1872            * PASS 1 PROCESSING
1873            *
1874            *
1875            * INDICATE THAT TOKENS ARE PRESENT IN THIS FILE.  PSEUDO-OP ID HA
1876            * TO KNOW.
1877            *
1878 EEB22 0000         CON(5) =LIT           16
          0
1879 EEB27 0100         CON(5) 16
          0
1880 EEB2C 0000         CON(5) =aSETF         SET.FLAG    ( * * )
          0
1881            *
1882            * SAVE LOWEST KNOWN TOKEN
1883            *
1884 EEB31 0000         CON(5) =aEXPR         EXPR
          0
1885 EEB36 0000         CON(5) =DUP           DUP         ( * NUM NUM * )
          0
1886 EEB3B 0000         CON(5) =aLOTOK        LOTOK
          0
1887 EEB40 0000         CON(5) =AT            @
          0
1888 EEB45 0000         CON(5) =MIN           MIN         ( * NUM MIN * )
          0
```



```
1889 EEB4A 0000         CON(5) =aLOTOK        LOTOK
          0
1890 EEB4F 0000         CON(5) =STORE         !           ( * NUM * )
          0
1891            *
1892            * SAVE HIGHEST KNOWN TOKEN
1893            *
1894 EEB54 0000         CON(5) =aHITOK        HITOK
          0
1895 EEB59 0000         CON(5) =AT            @
          0
1896 EEB5E 0000         CON(5) =MAX           MAX         ( * MAX * )
          0
1897 EEB63 0000         CON(5) =aHITOK        HITOK
          0
1898 EEB68 0000         CON(5) =STORE         !
          0
1899            *
1900            * RETURN LENGTH
1901            *
1902 EEB6D 0000         CON(5) =TWO           2           ( * NUM * )
          0
1903 EEB72 0000         CON(5) =SEMI          ;           ( * NUM * )
          0
1904            *
1905            * PASS 2 PROCESSING
1906            *
1907 EEB77               %IF136                ELSE        ( * * )
1908 EEB77 75DD         CON(5) =aENSLX        ENSURE.LEX  ( * * )
          E
1909 EEB7C FEAD         CON(5) =aWRUOL        WRAPUPOLD   ( * * )
          E
1910            *
1911            * GENERATE TWO NIBBLE CONSTANT OF TOKEN NUMBER
1912            *
1913 EEB81 0000         CON(5) =QUOTC         "   CON(2) "
          0
1914 EEB86 90           CON(2) 9
1915 EEB88 90           CON(2) 9
1916 EEB8A 0202         NIBASC \  \
1917 EEB8E 34F4         NIBASC \CON(2) \
          E482
          2392
          02
1918 EEB9C 0000         CON(5) =a>PAD         >PAD        ( * STR * )
          0
1919 EEBA1 0000         CON(5) =aOLDEX        OLDEXPR
          0
1920 EEBA6 0000         CON(5) =S<&           S<&         ( * STR * )
          0
1921            *
1922            * SIMULATE THIS LINE
1923            *
1924 EEBAB 62BD         CON(5) =aSTUNW        STARTUPNEW  ( * * )
          E
```

```
1925 EEBB0 0000         CON(5) =SEMI          ;           ( * * )
           0
1926            ****************************************************************
1927            ***
1928            ***  END.TEXT
1929            ***
1930            *** ( -- [NUM] )
1931            *** ( MARK THE END OF THE TEXT TABLE )
1932            ****************************************************************
1933 EEBB5 0000 =aENDTX CON(5) =DOCOL         : END.TEXT
           0
1934 EEBBA 0000         CON(5) =a?PAS1        ?PASS=1
           0
1935 EEBBF 0000         CON(5) =ZBRNH         IF
           0
1936 EEBC4 C300         REL(5) %IF157
           0
1937            *
1938            * PASS 1 PROCESSING
1939            *
1940 EEBC9 0000         CON(5) =aKENT         KENT
           0
1941 EEBCE 0000         CON(5) =AT            @
           0
1942 EEBD3 0000         CON(5) =ZEQ           0=
           0
1943 EEBD8 0000         CON(5) =ZBRNH         IF          ( * * )
           0
1944 EEBDD 9100         REL(5) %IF158
           0
1945            *
1946            * TEXT TABLE NOT PRESENT, WE MUST MAKE ONE (SINCE NO KEY PSEUDO-O
1947            * REACHED)
1948            *
1949 EEBE2 0000         CON(5) =aLC           LC
           0
1950 EEBE7 0000         CON(5) =AT            @           ( * NUM * )
           0
1951 EEBEC 139E         CON(5) =TxTbSt        TXTBST      ( * NUM STR * )
           E
1952 EEBF1 0000         CON(5) =a>SYT         >SYT        ( * * )
           0
1953            *
1954            * RETURN LENGTH
1955            *
1956 EEBF6       %IF158                       THEN        ( * * )
1957 EEBF6 0000         CON(5) =THREE         3           ( * NUM * )
           0
1958 EEBFB 0000         CON(5) =SEMI          ;           ( * NUM * )
           0
1959            *
1960            * PASS 2 PROCESSING
1961            *
1962 EEC00       %IF157                       ELSE        ( * * )
1963 EEC00 FEAD         CON(5) =aWRUOL        WRAPUPOLD   ( * * )
```



```
           E
1964 EEC05 0000         CON(5) =aKENT         KENT
           0
1965 EEC0A 0000         CON(5) =AT            @
           0
1966 EEC0F 0000         CON(5) =ZEQ           0=
           0
1967 EEC14 0000         CON(5) =ZBRNH         IF
           0
1968 EEC19 F000         REL(5) %IF159
           0
1969            *
1970            * GENERATE START OF TEXT TABLE
1971            *
1972 EEC1E 139E         CON(5) =TxTbSt        TXTBST      ( * STR * )
           E
1973 EEC23 1DAD         CON(5) =aSIMLN        SIM.LINE    ( * * )
           E
1974            *
1975            * GENERATE END OF TEXT TABLE
1976            *
1977 EEC28       %IF159                       THEN        ( * * )
1978 EEC28 0000         CON(5) =QUOTC         "   NIBHEX 1FF"
           0
1979 EEC2D C0           CON(2) 12
1980 EEC2F C0           CON(2) 12
1981 EEC31 0202         NIBASC \  NIBHEX\
           E494
           2484
           5485
1982 EEC41 0213         NIBASC \ 1FF\
           6464
1983            *
1984            * SIMULATE THIS LINE
1985            *
1986 EEC49 62BD         CON(5) =aSTUNU        STARTUPNEW  ( * * )
           E
1987 EEC4E 0000         CON(5) =SEMI          ;           ( * * )
           0
1988            ****************************************************************
1989            ***
1990            ***  BIN
1991            ***
1992            *** ( -- [NUM] )
1993            *** ( CREATE A BIN FILE HEADER )
1994            ****************************************************************
1995 EEC53 0000 =aBIN   CON(5) =DOCOL         : BIN
           0
1996            *
1997            * CHECK FOR FILETYPE & MULTIPLE BIN PSEUDO-OPS
1998            *
1999 EEC58 CBCD         CON(5) =aFILBN        FILE.IS.BIN
           E
2000 EEC5D 0000         CON(5) =a?PAS1        ?PASS=1
           0
```

```
2001 EEC62 00000         CON(5) =ZBRNH         IF
           0
2002 EEC67 B4000         REL(5) %IF137
           0
2003               *
2004               * PASS 1
2005               *
2006               *
2007               * CHECK FOR VALID FILENAME & PURGE PREVIOUS BIN FILE
2008               *
2009 EEC6C 00000         CON(5) =aEXP.F        EXPRESSION.FIELD
           0
2010 EEC71 602E          CON(5) =aVALID        VALID.NAME  ( * STR * )
           E
2011 EEC76 00000         CON(5) =LIT           bin file type
           0
2012 EEC7B 402E          CON(5) #E204
           0
2013 EEC80 00000         CON(5) =aPURGE        PURGE       ( * F * )
           0
2014 EEC85 00000         CON(5) =DROP          DROP        ( * * )
           0
2015               *
2016               * UPDATE POINTERS TO REFLECT MEMORY MOVE
2017               *
2018 EEC8A 00000         CON(5) =aMEMOV        MEMORY MOVED
           0
2019 EEC8F 00000         CON(5) =aOBJFL        OBJ.FILE
           0
2020 EEC94 00000         CON(5) =AT            @
           0
2021 EEC99 00000         CON(5) =aLC           LC
           0
2022 EEC9E 00000         CON(5) =STORE         !
           0
2023               *
2024               * RETURN LENGTH
2025               *
2026 EECA3 00000         CON(5) =LIT           37          ( * NUM * )
           0
2027 EECA8 52000         CON(5) 37
           0
2028 EECAD 00000         CON(5) =SEMI          ;           ( * NUM * )
           0
2029 EECB2       %IF137
2030               *
2031               * PASS 2
2032               *
2033               *
2034               * CHECK FOR VALID NAME & OUTPUT NAME LINE
2035               *
2036 EECB2 00000         CON(5) =aOLDEX        OLD.EXPR
           0
2037 EECB7 602E          CON(5) =aVALID        VALID.NAME  ( * STR * )
           E
```



```
2038 EECBC 742E          CON(5) =aNAMEF        NAMEFIELD   ( * * )
           E
2039               *
2040               * OUTPUT FILE TYPE LINE
2041               *
2042 EECC1 00000         CON(5) =QUOTC         "   NIBHEX 402E"
           0
2043 EECC6 D0            CON(2) 13
2044 EECC8 D0            CON(2) 13
2045 EECCA 0202          NIBASC \  \
2046 EECCE E494          NIBASC \NIBHEX 4\
           2484
           5485
           0243
2047 EECDE 03235         NIBASC \02E\
           4
2048 EECE4 1DAD          CON(5) =aSIMLN        SIM.LINE    ( * * )
           E
2049               *
2050               * OUTPUT FLAGS LINE
2051               *
2052 EECE9 00000         CON(5) =QUOTC         "   CON(2) 0"
           0
2053 EECEE A0            CON(2) 10
2054 EECF0 A0            CON(2) 10
2055 EECF2 0202          NIBASC \  \
2056 EECF6 34F4          NIBASC \CON(2) 0\
           E482
           2392
           0203
2057 EED06 1DAD          CON(5) =aSIMLN        SIM.LINE    ( * * )
           E
2058               *
2059               * OUTPUT TIME AND DATE LINES
2060               *
2061 EED0B EE2E          CON(5) =aTI/DA        TIME/DATE   ( * * )
           E
2062               *
2063               * OUTPUT FILE CHAIN LINK LINE
2064               *
2065 EED10 00000         CON(5) =QUOTC         "   REL(5) FiLeNd"
           0
2066 EED15 F0            CON(2) 15
2067 EED17 F0            CON(2) 15
2068 EED19 0202          NIBASC \  \
2069 EED1D 2554          NIBASC \REL(5) F\
           C482
           5392
           0264
2070 EED2D 96C4          NIBASC \iLeNd\
           56E4
           46
2071 EED37 62BD          CON(5) =aSTUNW        STARTUPNEW  ( * * )
           E
2072 EED3C 00000         CON(5) =SEMI          ;           ( * * )
```

```
          0
2073            ****************************************************************
2074            ***
2075            ***  CHAIN
2076            ***
2077            *** ( -- )
2078            *** ( APPEND BINARY SUBHEADER TO HEADER )
2079            ****************************************************************
2080 EED41 0000 =aCHAIN CON(5) =DOCOL          : CHAIN
          0
2081 EED46 B6DD         CON(5) =aENSBN         ENSURE.BIN  ( * * )
          E
2082 EED4B FEAD         CON(5) =aWRUOL         WRAPUPOLD   ( * * )
          E
2083 EED50 0000         CON(5) =aEXPR          EXPR
          0
2084 EED55 0000         CON(5) =aNEG1          -1
          0
2085 EED5A 0000         CON(5) =EQUAL          =
          0
2086 EED5F 0000         CON(5) =ZBRNH          IF          ( * * )
          0
2087 EED64 A200         REL(5) %IF138
          0
2088            *
2089            * USER INDICATED NO CHAIN (-1).
2090            *
2091 EED69 0000         CON(5) =QUOTC          "   CON(5) "
          0
2092 EED6E 90           CON(2) 9
2093 EED70 90           CON(2) 9
2094 EED72 0202         NIBASC \  \
2095 EED76 34F4         NIBASC \CON(5) \
          E482
          5392
          02
2096 EED84 0000         CON(5) =BRNCH
          0
2097 EED89 0200         REL(5) %1H138
          0
2098            *
2099            * USER INDICATED A LABEL FOR SUB PROGRAM CHAIN
2100            *
2101 EED8E       %IF138                        ELSE        ( * * )
2102 EED8E 0000         CON(5) =QUOTC          "   REL(5) "
          0
2103 EED93 90           CON(2) 9
2104 EED95 90           CON(2) 9
2105 EED97 0202         NIBASC \  \
2106 EED9B 2554         NIBASC \REL(5) \
          C482
          5392
          02
2107            *
2108            * SIMULATE THIS LINE
```



```
2109            *
2110 EEDA9       %1H138                        THEN        ( * STR * )
2111 EEDA9 0000         CON(5) =a>PAD          >PAD
          0
2112 EEDAE 0000         CON(5) =aOLDEX         OLDEXPR
          0
2113 EEDB3 0000         CON(5) =S<&            S<&         ( * STR * )
          0
2114 EEDB8 1DAD         CON(5) =aSIMLN         SIM.LINE    ( * * )
          E
2115            *
2116            * GENERATE NO LABEL CHAIN IN THIS FILE
2117            *
2118 EEDBD 0000         CON(5) =QUOTC          "   CON(5) -1"
          0
2119 EEDC2 B0           CON(2) 11
2120 EEDC4 B0           CON(2) 11
2121 EEDC6 0202         NIBASC \  \
2122 EEDCA 34F4         NIBASC \CON(5) -\
          E482
          5392
          02D2
2123 EEDDA 13           NIBASC \1\                            ( * STR * )
2124 EEDDC 1DAD         CON(5) =aSIMLN         SIM.LINE    ( * * )
          E
2125            *
2126            * GENERATE TWO NIBBLE FILLER
2127            *
2128 EEDE1 0000         CON(5) =QUOTC          "   NIBHEX 20"
          0
2129 EEDE6 B0           CON(2) 11
2130 EEDE8 B0           CON(2) 11
2131 EEDEA 0202         NIBASC \  \
2132 EEDEE E494         NIBASC \NIBHEX 2\
          2484
          5485
          0223
2133 EEDFE 03           NIBASC \0\                            ( * STR * )
2134            *
2135            * SIMULATE THIS LINE
2136            *
2137 EEE00 62BD         CON(5) =aSTUNW         STARTUPNEW  ( * * )
          E
2138 EEE05 0000         CON(5) =SEMI           ;           ( * * )
          0
```

```
2139                    $TITLE FIT STATS
2140            zSIZE   EQU    (*)-zTHIS
2141            zLEFT   EQU    (zNEXT)-*
2142 EEE0A              BSS    zLEFT
2143 EEF00              END
```



```
%BG13   Abs  973376 #EDA40 -   329   341
%BG14   Abs  974401 #EDE41 -   786   849
%DO12   Abs  972669 #ED77D -    65    72
%DO13   Abs  974956 #EE06C -   994   997
%GOGET  Abs  975568 #EE2D0 -  1230  1216
%IF104  Abs  972579 #ED723 -    36    27
%IF105  Abs  972734 #ED7BE -    95    87
%IF106  Abs  974287 #EDDCF -   737   729
%IF107  Abs  973162 #ED96A -   244   235
%IF108  Abs  973212 #ED99C -   262   251
%IF109  Abs  973456 #EDA90 -   350   320
%IF111  Abs  973942 #EDC76 -   579   569
%IF112  Abs  974142 #EDD3E -   667   660
%IF115  Abs  974062 #EDCEE -   632   600   620
%IF117  Abs  974446 #EDE6E -   799   792
%IF118  Abs  974511 #EDEAF -   822   815
%IF119  Abs  974816 #EDFE0 -   955   939
%IF121  Abs  974881 #EE021 -   972   967
%IF122  Abs  974971 #EE07B -   998   991
%IF123  Abs  975021 #EE0AD -  1012  1007
%IF124  Abs  975181 #EE14D -  1060  1058
%IF125  Abs  975216 #EE170 -  1081  1072
%IF126  Abs  975306 #EE1CA -  1121  1112
%IF127  Abs  975426 #EE242 -  1169  1162
%IF128  Abs  976028 #EE49C -  1366  1332
%IF129  Abs  976206 #EE54E -  1430  1420
%IF130  Abs  976634 #EE6FA -  1540  1526
%IF131  Abs  976755 #EE773 -  1584  1567
%IF132  Abs  976914 #EE812 -  1629  1613
%IF133  Abs  977372 #EE9DC -  1784  1740
%IF134  Abs  977297 #EE991 -  1759  1748
%IF135  Abs  977497 #EEA59 -  1809  1791
%IF136  Abs  977783 #EEB77 -  1907  1870
%IF137  Abs  978098 #EECB2 -  2029  2002
%IF138  Abs  978318 #EED8E -  2101  2087
%IF142  Abs  973049 #ED8F9 -   201   121
%IF143  Abs  973089 #ED921 -   217   137   190
%IF144  Abs  972904 #ED868 -   158   129
%IF145  Abs  972969 #ED8A9 -   178   166
%IF146  Abs  973074 #ED912 -   210   204
%IF147  Abs  973706 #EDB8A -   490   479
%IF148  Abs  973721 #EDB99 -   497   471
%IF149  Abs  973857 #EDC21 -   549   514
%IF150  Abs  973957 #EDC85 -   586   552
%IF151  Abs  972619 #ED74B -    51    40
%IF152  Abs  976342 #EE5D6 -  1463  1457
%IF153  Abs  976426 #EE62A -  1483  1477
%IF156  Abs  974082 #EDD02 -   644   635
%IF157  Abs  977920 #EEC00 -  1962  1936
%IF158  Abs  977910 #EEBF6 -  1956  1944
%IF159  Abs  977960 #EEC28 -  1977  1968
%RP13   Abs  973436 #EDA7C -   342   337
%TH117  Abs  974571 #EDEEB -   847   798
%TH123  Abs  975026 #EE0B2 -  1014  1011
%TH130  Abs  976663 #EE717 -  1552  1539
```

```
%TH131  Abs  976784 #EE790 -  1593  1580
%TH138  Abs  978345 #EEDA9 -  2110  2097
2DROP   Ext                -   170   239   968  1013
2DUP    Ext                -   228   834  1158  1182  1833
>R      Ext                -   287   804   830  1003
?DUP    Ext                -   318   989
ABS     Ext                -    31    44
ADD     Ext                -   152   162   184   289   741   836
ALLOT   Ext                -   582
AND     Ext                -   119
AT      Ext                -   185   511   565   581   597   617   656   719
                             927   935   944   951   953   963   964  1357
                            1460  1480  1610  1644  1713  1745  1753  1767
                            1788  1887  1895  1941  1950  1965  2020
BL      Ext                -   332
BRNCH   Ext                -   340   797  1010  1215  1538  1579  2096
C@+     Ext                -   338
CAT     Ext                -   331
CHR$    Ext                -  1026
CONTX   Ext                -   945   962
CURR    Ext                -   943   952
DEC     Ext                -   764
DOCOL   Ext                -    23    84   110   226   275   304   360   373
                             385   407   448   467   509   595   615   654
                             676   687   697   708   750   778   861   875
                             914  1038  1054  1069  1109  1148  1178  1210
                            1225  1243  1325  1417  1521  1562  1603  1674
                            1700  1724  1737  1867  1933  1995  2080
DROP    Ext                -   116   282   339   722   819   960   979  1233
                            1344  1777  1837  2014
DUP     Ext                -    37    52   159   163   286   540   726   808
                             812  1004  1885
END$    Ext                -   325  1232
EQUAL   Ext                -   187   333   657   937  2085
F.AND   Ext                -   335
FIND>   Ext                -   965
FIVE+   Ext                -   321
FSTX    Ext                -  1159
GT      Ext                -   567
HERE    Ext                -   539   559
HEX     Ext                -   762
LATE    Ext                -   761
LIT     Ext                -    45    58   150   160   171   182   194   211
                             263   346   491   561   573   664   730   739
                             788   809   820   969  1166  1254  1283  1298
                            1341  1363  1421  1453  1473  1878  2011  2026
LT      Ext                -   790
LTZ     Ext                -    38
MAX     Ext                -    62  1896
MIN     Ext                -    60   348   720   928  1888
MINUS   Ext                -   295   563   566   828
MOD     Ext                -   811
N!      Ext                -   153   181
N@      Ext                -   164
NULL$   Ext                -   533
```



```
ONE     Ext                -    61   518   677   782   936  1269  1313  1630
                            1760  1810
ONE+    Ext                -    68   803  1779
ONE-    Ext                -   988  1005  1008  1839
OVER    Ext                -   330   334   787   827
POS     Ext                -   314
PSTOR   Ext                -  1632  1762  1812
QUOTC   Ext                -   229   245   308   524   752   862   884  1019
                            1039  1090  1130  1188  1211  1226  1248  1263
                            1277  1292  1307  1379  1389  1402  1436  1448
                            1468  1488  1497  1506  1530  1544  1571  1585
                            1617  1636  1647  1657  1680  1704  1725  1795
                            1821  1846  1913  1978  2042  2052  2065  2091
                            2102  2118  2128
R>      Ext                -   294   837   839  1025
R@      Ext                -    67
ROT     Ext                -   324  1231
S!      Ext                -   363   376   390   427   456   535   919   930
                            1184  1835
S0      Ext                -   560
S<&     Ext                -   765   866   892   897   981  1024  1027  1195
                            1253  1257  1268  1271  1282  1286  1297  1301
                            1312  1315  1443  1462  1482  1537  1578  1646
                            1652  1664  1687  1715  1841  1853  1920  2113
S=      Ext                -   233   249
SEMI    Ext                -    32    76   102   154   174   197   206   218
                             243   258   266   296   349   352   365   377
                             399   440   459   486   499   545   587   607
                             627   646   668   679   690   700   742   770
                             853   867   906  1030  1046  1061  1077  1100
                            1117  1140  1170  1201  1235  1317  1365  1409
                            1426  1513  1554  1595  1666  1692  1716  1729
                            1780  1859  1903  1925  1958  1987  2028  2072
                            2138
STORE   Ext                -    54    70   101   242   257   421   520   542
                             544   606   626   946   954  1351  1359  1890
                            1898  2022
SWAP    Ext                -   115   134   180   279   281   290   721   826
                             829   978  1776  1836
SWAP2   Ext                -   838   896  1194
THREE   Ext                -   624   698  1957
TWO     Ext                -   179   475   483   604   688  1234  1902
TWO*    Ext                -   288   738   835  1778  1838
TWO-    Ext                -   961
=TxTbSt Abs  977201 #EE931 -  1724  1754  1804  1951  1972
WIDTH   Ext                -   718   926
XDO     Ext                -    64   993
XLOOP   Ext                -    71   996
ZBRNH   Ext                -    26    39    86   120   128   136   165   189
                             203   234   250   319   336   470   478   513
                             551   568   599   619   634   659   728   791
                             814   848   938   966   990  1006  1057  1071
                            1111  1161  1331  1419  1456  1476  1525  1566
                            1612  1739  1747  1790  1869  1935  1943  1967
                            2001  2086
```

```
 ZEQ      Ext                  -   135   188   469   477   512   598   618   658
                                   727   813  1160  1611  1746  1789  1942  1966
 ZERO     Ext                  -    63    66    91   133   147   205   255   419
                                   498   843   992  1059  1349
 a&CLR    Ext                  -   398   439   879
=a+LAb    Abs  973232 #ED9B0   -   275   313   323
 a>OPx    Ext                  -    69
 a>PAD    Ext                  -   757  1193  1441  1458  1478  1535  1576  1642
                                  1662  1685  1708  1826  1851  1918  2111
 a>SYT    Ext                  -   143  1755  1769  1952
=a?1ST    Abs  973636 #EDB44   -   467   550   633
 a?PAS1   Ext                  -    25    85   127   202   468  1056  1070  1110
                                  1330  1418  1738  1868  1934  2000
 aBAS$    Ext                  -  1230
=aBIN     Abs  978003 #EEC53   -  1995
 aBLNK1   Ext                  -   351
=aBRK8    Abs  974391 #EDE37   -   778   987
=aBSS     Abs  972544 #ED700   -    23
=aCHAIN   Abs  978241 #EED41   -  2080
=aCHAR    Abs  977082 #EE8BA   -  1674
=aCLCWD   Abs  974207 #EDD7F   -   708  1076  1116
=aDATE$   Abs  975544 #EE2B8   -  1225  1285  1300  1314
 aDOLIN   Ext                  -   394
 aDONE    Ext                  -   100
 aEJECT   Ext                  -   364
=aEND     Abs  972709 #ED7A5   -    84
=aENDQ    Abs  974586 #EDEFA   -   861   901  1028  1196  1854
=aENDTX   Abs  977845 #EEBB5   -  1933
=aENS4T   Abs  974147 #EDD43   -   676  1082  1122
 aENS8L   Ext                  -  1154
=aENSBN   Abs  974187 #EDD6B   -   697  2081
=aENSLX   Abs  974167 #EDD57   -   687  1431  1522  1563  1604  1675  1785  1908
=aENTRY   Abs  976794 #EE79A   -  1603
=aEQU     Abs  972754 #ED7D2   -   110
 aERROR   Ext                  -    47   173   196   213   265   493   666   971
 aEXP.F   Ext                  -   227   425   712  1339  1773  2009
 aEXPR    Ext                  -    24   141   186  1524  1565  1884  2083
=aFIL4T   Abs  973731 #EDBA3   -   509  1055
=aFILBN   Abs  974012 #EDCBC   -   615  1999
=aFILLX   Abs  973962 #EDC8A   -   595  1329
 aFLSIZ   Ext                  -   564   580
=aFLTST   Abs  974092 #EDD0C   -   654   678   689   699
 aFLTYP   Ext                  -   510   519   596   605   616   625   655   934
=aFORTH   Abs  975151 #EE12F   -  1054
=aGENIx   Abs  977149 #EE8FD   -  1700  1768  1816
 aHITOK   Ext                  -  1479  1894  1897
=aIDN     Abs  976171 #EE52B   -  1417
 aKENT    Ext                  -  1712  1744  1761  1787  1811  1940  1964
=aKEYP    Abs  977232 #EE950   -  1737
 aKILL    Ext                  -   528
 aLAB.F   Ext                  -   114   117   125   142   148   280
 aLABOK   Ext                  -   118
 aLC      Ext                  -   543   950  1358  1752  1766  1949  2021
 aLC+     Ext                  -   435
=aLEX     Abs  975918 #EE42E   -  1325
```



```
 aLINEC   Ext                  -  1350
=aLIST    Abs  973094 #ED926   -   226
 aLLINE   Ext                  -   880  1018  1247  1262  1276  1291  1306
 aLOTOK   Ext                  -  1459  1886  1889
 aLSTCH   Ext                  -  1002
 aLUKUP   Ext                  -   126   149
 aMEMOV   Ext                  -   529  1355  2018
=aMSG     Abs  976552 #EE6A8   -  1521
=aNAMEF   Abs  975431 #EE247   -  1178  1375  2038
 aNEG1    Ext                  -    99   240   485   796  2084
 aNUM#2   Ext                  -  1645  1714  1840
 aNUM#3   Ext                  -   980  1461  1481
 aNUM#5   Ext                  -   763
 aOBJFL   Ext                  -   541  1356  2019
 aOBJGR   Ext                  -   640
 aOFILE   Ext                  -   534  1183
 aOLDEX   Ext                  -   426   923   929   959   977   986  1373  1442
                                  1536  1577  1663  1686  1827  1834  1852  1919
                                  2036  2112
 aOPLEN   Ext                  -    53   420
 aPARSE   Ext                  -   457
=aPOLL    Abs  976673 #EE721   -  1562
 aPROCS   Ext                  -   458
 aPURGE   Ext                  -  1343  2013
 aQ"STR   Ext                  -   713   924  1152  1774  1828
 aSETF    Ext                  -   484  1880
 aSHUTD   Ext                  -   575   732  1168
=aSIMFF   Abs  974612 #EDF14   -   875   995  1009
=aSIMLN   Abs  973521 #EDAD1   -   385   769   905   982  1029  1200  1258  1272
                                  1287  1302  1316  1385  1394  1444  1464  1484
                                  1493  1502  1625  1653  1803  1805  1817  1842
                                  1973  2048  2057  2114  2124
 aSLINE   Ext                  -   312   322   389   455   918   976
 aSPIT    Ext                  -   431
 aSTIT    Ext                  -   362
=aSTITL   Abs  973466 #EDA9A   -   360
=aSTUNW   Abs  973606 #EDB26   -   448  1045  1408  1512  1553  1594  1665  1691
                                  1858  1924  1986  2071  2137
 aTENT    Ext                  -  1609  1631  1643
 aTESTF   Ext                  -   476  1455  1475
=aTI/DA   Abs  975598 #EE2EE   -  1243  1398  2061
=aTIME$   Abs  975510 #EE296   -  1210  1256  1270
 aTIT     Ext                  -   375
=aTITLE   Abs  973496 #EDAB8   -   373
=aTOKNP   Abs  977678 #EEB0E   -  1867
 aTOLST   Ext                  -   241   256
 aTRIM    Ext                  -   714   925  1153  1775  1829
=aUTCFA   Abs  975103 #EE0FF   -  1038  1099  1139
=aUTLFA   Abs  974312 #EDDE8   -   750  1086  1126
=aUTNFA   Abs  974686 #EDF5E   -   914  1095  1135
=aVALID   Abs  975366 #EE206   -  1148  1340  1374  2010  2037
=aWORDI   Abs  975276 #EE1AC   -  1109
=aWORDN   Abs  975186 #EE152   -  1069
=aWRUOL   Abs  973551 #EDAEF   -   407   645   751  1432  1523  1564  1605  1676
                                  1786  1909  1963  2082
```

```
=aXTRCT  Abs  973297 #ED9F1 -   304   361   374
 zLEFT   Abs     246 #000F6 -  2141  2142
 zNEXT   Abs  978688 #EEF00 -     5  2141
 zSIZE   Abs    5898 #0170A -  2140
 zTHIS   Abs  972544 #ED700 -     4  2140
```


Input Parameters

Source file name is GN&AS6

Listing file name is GN/AS6

Object file name is GN%AS6::65

```
                                111111
                      0123456789012345
```

Initial flag settings are

Errors

None

Saturn Assembler News

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO CONTACT THE MANUFACTURER


```
 1                      TITLE  AS7:_FORTH_ASSEMBLER_PARSING
 2                      RDSYMB MR%GTO
 3 EEF00                ABS    #EEF00
 4              zTHIS   EQU    *
 5              zNEXT   EQU    #EF600
 6
 7              ********************************************************
 8              ***
 9              *** MR&AS7      <840504.1037>
10              ***
11              ***
12              ***  FORTH ASSEMBLER
13              ***  PARSING OF SOURCE LINES
14              ***
15              ********************************************************
16              ****************************************************************
17              ***
18              ***  HASH
19              ***
20              *** ( STR -- NUM )
21              ****************************************************************
22 EEF00 0000  =aHASH  CON(5) =DOCOL          : HASH
       0
23              *
24              * SET UP COUNT
25              *
26 EEF05 0000          CON(5) =ZERO           0
       0
27 EEF0A 0000          CON(5) =LIT            6
       0
28 EEF0F 6000          CON(5) 6
       0
29 EEF14 0000          CON(5) =ZERO           0
       0
30 EEF19 0000          CON(5) =XDO            DO          ( * STR NUM * )
       0
31              *
32              * GO THROUGH EACH CHARACTER & EXTRACT LOW ORDER THREE BITS
33              *
34 EEF1E        %DO14
35 EEF1E 0000          CON(5) =>R             >R
       0
36 EEF23 0000          CON(5) =C@+            C@+
       0
37 EEF28 0000          CON(5) =LIT            7
       0
38 EEF2D 7000          CON(5) 7
       0
39 EEF32 0000          CON(5) =AND            AND         ( * STR C * )
       0
40 EEF37 0000          CON(5) =R>             R>
       0
41              *
42              * SHIFT COUNT THREE BITS AND ADD IN BITS FROM CHARACTER
43              *
```

```
 44 EEF3C 0000         CON(5) =LIT          8
          0
 45 EEF41 8000         CON(5) 8
          0
 46 EEF46 0000         CON(5) =MULT         *
          0
 47 EEF4B 0000         CON(5) =ADD          +          ( * STR NUM * )
          0
 48 EEF50 0000         CON(5) =XLOOP        LOOP       ( * STR NUM * )
          0
 49 EEF55 9CFF         REL(5) %D014
          F
 50             *
 51             * DROP THE STRING
 52             *
 53 EEF5A 0000         CON(5) =>R           >R
          0
 54 EEF5F 0000         CON(5) =2DROP        2DROP
          0
 55 EEF64 0000         CON(5) =R>           R>
          0
 56             *
 57             * MOD THE RESULT BY 43
 58             *
 59 EEF69 0000         CON(5) =LIT          43
          0
 60 EEF6E B200         CON(5) 43
          0
 61 EEF73 0000         CON(5) =MOD          MOD
          0
 62 EEF78 0000         CON(5) =SEMI         ;          ( * NUM * )
          0
 63             ****************************************************************
 64             ***
 65             ***  JUMP
 66             ***
 67             *** ( NUM -- PTR )
 68             ****************************************************************
 69 EEF7D 0000 =aJUMP  CON(5) =DOCOL        : JUMP
          0
 70             *
 71             * GET TO APPROPRIATE PLACE IN OPCODE TABLE (PART OF AS0)
 72             *
 73 EEF82 0000         CON(5) =LIT          5
          0
 74 EEF87 5000         CON(5) 5
          0
 75 EEF8C 0000         CON(5) =MULT         *
          0
 76 EEF91 0000         CON(5) =LIT          OPCODE TABLE
          0
 77 EEF96 0000         CON(5) =OPTABL
          0
 78 EEF9B 0000         CON(5) =ADD          +
          0
```



```
 79 EEFA0 0000         CON(5) =AT           @
          0
 80 EEFA5 0000         CON(5) =SEMI         ;          ( * PTR * )
          0
 81             ****************************************************************
 82             ***
 83             ***  SEARCH
 84             ***
 85             *** ( STR PTR -- PTR TF )
 86             *** ( STR PTR -- FF )
 87             ****************************************************************
 88 EEFAA 0000 =aSRCH  CON(5) =DOCOL        : SEARCH
          0
 89             *
 90             * SAVE TOP OF THIS OPCODE GROUP & INITIALIZE WHICH ENTRY IN GROUP
 91             *
 92 EEFAF 0000         CON(5) =aOPCTO       OPCTOP
          0
 93 EEFB4 0000         CON(5) =STORE        !
          0
 94 EEFB9 0000         CON(5) =ONE          1
          0
 95 EEFBE 0000         CON(5) =aOPCPO       OPCPOS
          0
 96 EEFC3 0000         CON(5) =STORE        !          ( * STR * )
          0
 97 EEFC8       %BG15                       BEGIN      ( * STR * )
 98             *
 99             * STORE NEXT OPCODE POSITION
100             *
101 EEFC8 0000         CON(5) =aOPCPO       OPCPOS
          0
102 EEFCD 0000         CON(5) =AT           @
          0
103 EEFD2 0000         CON(5) =DUP          DUP        ( * STR POS POS * )
          0
104 EEFD7 0000         CON(5) =TWO*         2*
          0
105 EEFDC 0000         CON(5) =aOPCPO       OPCPOS
          0
106 EEFE1 0000         CON(5) =STORE        !          ( * STR POS * )
          0
107             *
108             * CALCULATE ADDRESS OF THIS OPCODE POSITION
109             *
110 EEFE6 0000         CON(5) =ONE-         1-
          0
111 EEFEB 0000         CON(5) =LIT          25
          0
112 EEFF0 9100         CON(5) 25
          0
113 EEFF5 0000         CON(5) =MULT         *          ( * STR POS * )
          0
114 EEFFA 0000         CON(5) =aOPCTO       OPCTOP
          0
```

```
115 EEFFF 0000          CON(5) =AT              @
          0
116 EF004 0000          CON(5) =ADD             +               ( * STR POS * )
          0
117             *
118             * SAVE CURRENT POSITION
119             *
120 EF009 0000          CON(5) =DUP             DUP
          0
121 EF00E 0000          CON(5) =aOPCPT          OPCPTR
          0
122 EF013 0000          CON(5) =STORE           !               ( * STR PTR * )
          0
123             *
124             * GET OPCODE TEXT
125             *
126 EF018 0000          CON(5) =THREE           3
          0
127 EF01D 0000          CON(5) =ADD             +
          0
128 EF022 0000          CON(5) =DUP             DUP
          0
129 EF027 0000          CON(5) =ONE-            1-
          0
130 EF02C 0000          CON(5) =N@              N@              ( * STR STR' * )
          0
131             *
132             * CHECK IF WE'VE FOUND OUR OPCODE
133             *
134 EF031 0000          CON(5) =aDUP4           4DUP
          0
135 EF036 0000          CON(5) =S=              S=
          0
136 EF03B 0000          CON(5) =ZBRNH           IF              ( * STR STR' * )
          0
137 EF040 E100          REL(5) %IF139
          0
138 EF045 0000          CON(5) =aDROP4          4DROP
          0
139 EF04A 0000          CON(5) =aNEG1           -1
          0
140 EF04F 0000          CON(5) =aNEG1           -1              ( * TF TF * )
          0
141 EF054 0000          CON(5) =BRNCH           ELSE            ( * STR STR' * )
          0
142 EF059 D700          REL(5) %TH139
          0
143 EF05E       %IF139
144             *
145             * CHECK IF END OF TREE
146             *
147 EF05E 0000          CON(5) =aOPCPT          OPCPTR
          0
148 EF063 0000          CON(5) =AT              @
          0
```



```
149 EF068 0000          CON(5) =LIT             15
          0
150 EF06D F000          CON(5) 15
          0
151 EF072 0000          CON(5) =ADD             +
          0
152 EF077 0000          CON(5) =CAT             C@              ( * STR STR' FLGS * )
          0
153 EF07C 0000          CON(5) =LIT             4
          0
154 EF081 4000          CON(5) 4
          0
155 EF086 0000          CON(5) =AND             AND
          0
156 EF08B 0000          CON(5) =ZBRNH           IF              ( * STR STR' * )
          0
157 EF090 E100          REL(5) %IF140
          0
158             *
159             * WE'VE REACHED THE END OF THE TREE & HAVE NOT FOUND OUR OPCODE
160             *
161 EF095 0000          CON(5) =aDROP4          4DROP
          0
162 EF09A 0000          CON(5) =ZERO            0
          0
163 EF09F 0000          CON(5) =aNEG1           -1              ( * FF TF * )
          0
164 EF0A4 0000          CON(5) =BRNCH           ELSE            ( * STR STR' * )
          0
165 EF0A9 D200          REL(5) %TH140
          0
166 EF0AE       %IF140
167             *
168             * TREE STILL HAS ENTRIES, SHOULD WE GO LEFT OR RIGHT ?
169             *
170 EF0AE 0000          CON(5) =OVER2           2OVER
          0
171 EF0B3 0000          CON(5) =S<              S<
          0
172 EF0B8 0000          CON(5) =ZBRNH           IF              ( * STR * )
          0
173 EF0BD 4100          REL(5) %IF141
          0
174             *
175             * RIGHT
176             *
177 EF0C2 0000          CON(5) =ONE             1
          0
178 EF0C7 0000          CON(5) =aOPCPO          OPCPOS
          0
179 EF0CC 0000          CON(5) =PSTOR           +!              ( * STR * )
          0
180 EF0D1       %IF141                          THEN
181 EF0D1 0000          CON(5) =ZERO            0               ( * STR FF * )
          0
```

```
182 EF0D6       %TH140                      THEN
183 EF0D6       %TH139                      THEN        ( * STR FF / F TF * )
184 EF0D6 0000          CON(5) =ZBRNH       UNTIL
          0
185 EF0DB DEEF          REL(5) %BG15
          F
186             *
187             * HAVE WE FOUND OURS ?
188             *
189 EF0E0 0000          CON(5) =DUP         DUP
          0
190 EF0E5 0000          CON(5) =ZBRNH       IF          ( * F * )
          0
191 EF0EA 4100          REL(5) %IF142
          0
192             *
193             * RETURN ADDRESS OF WHAT WE FOUND
194             *
195 EF0EF 0000          CON(5) =aOPCPT      OPCPTR
          0
196 EF0F4 0000          CON(5) =AT          @
          0
197 EF0F9 0000          CON(5) =SWAP        SWAP
          0
198 EF0FE       %IF142                      THEN
199 EF0FE 0000          CON(5) =SEMI        ;           ( * [PTR] F * )
          0
200             ****************************************************************
201             ***
202             ***  OPC.LOOKUP
203             ***
204             *** ( STR -- PTR TF )
205             *** ( STR -- FF )
206             *** ( LOOKUP THE OPCODE IN STR IN THE OPCODE TABLES )
207             *** ( RETURN A POINTER TO THE ENTRY IF FOUND )
208             ****************************************************************
209 EF103 0000 =aOPLUK CON(5) =DOCOL       : OPC.LOOKUP
          0
210             *
211             * DOES THE HOOK ADDRESS FOR THE PARSE ROUTINE HAVE AN ADDRESS IN
212             *
213 EF108 0000          CON(5) =LIT         PARSE RT
          0
214 EF10D 97CF          CON(5) =oPARRT
          2
215 EF112 0000          CON(5) =AT          @           ( * STR PTR * )
          0
216 EF117 0000          CON(5) =?DUP        ?DUP
          0
217 EF11C 0000          CON(5) =ZBRNH       IF          ( * STR [PTR] * )
          0
218 EF121 3200          REL(5) %IF303
          0
219             *
220             * YES IT DOES, LET IT HAVE A CHANCE AT THIS OPCODE
```



```
221             *
222 EF126 0000          CON(5) =EXEC        EXECUTE     ( * [PTR] F * )
          0
223 EF12B 0000          CON(5) =ZBRNH       IF
          0
224 EF130 F000          REL(5) %IF150
          0
225             *
226             * IT HANDLED IT !
227             *
228 EF135 0000          CON(5) =aNEG1       -1          ( * PTR TF * )
          0
229 EF13A 0000          CON(5) =SEMI        ;           ( * PTR TF * )
          0
230 EF13F       %IF150                      ELSE        ( * * )
231             *
232             * WE'VE GOT TO LOOK FOR THIS OPCODE
233             *
234 EF13F 0000          CON(5) =aOPC.F      OPCODE.FIELD
          0
235 EF144       %IF303                                  ( * STR * )
236 EF144 0000          CON(5) =2DUP        2DUP        ( * STR STR * )
          0
237 EF149 00FE          CON(5) =aHASH       HASH        ( * STR NUM * )
          E
238 EF14E D7FE          CON(5) =aJUMP       JUMP        ( * STR PTR * )
          E
239 EF153 AAFE          CON(5) =aSRCH       SEARCH      ( * [PTR] F * )
          E
240 EF158 0000          CON(5) =SEMI        ;           ( * [PTR] F * )
          0
241             ****************************************************************
242             ***
243             ***  SET.OP.VARS
244             ***
245             *** ( PTR -- )
246             *** ( TAKE THE POINTER FROM OPC.LOOKUP AND FILL IN )
247             *** ( THE VARIABLES ASSOCIATED WITH A OPCODE LOOKUP )
248             ****************************************************************
249 EF15D 0000 =aSTVAR CON(5) =DOCOL       : SET.OP.VARS
          0
250 EF162 0000          CON(5) =DUP         DUP
          0
251             *
252             * GET OPCODE CLASS
253             *
254 EF167 0000          CON(5) =CAT         C@
          0
255 EF16C 0000          CON(5) =aOPTYP      OPTYPE
          0
256 EF171 0000          CON(5) =STORE       !
          0
257             *
258             * GET OPCODE FLAGS
259             *
```

```
260 EF176 0000         CON(5) =LIT          15
          0
261 EF17B F000         CON(5) 15
          0
262 EF180 0000         CON(5) =ADD          +          ( * PTR * )
          0
263 EF185 0000         CON(5) =DUP          DUP
          0
264 EF18A 0000         CON(5) =CAT          C@
          0
265 EF18F 0000         CON(5) =aOPFLG       OPFLAGS
          0
266 EF194 0000         CON(5) =STORE        !
          0
267             *
268             * GET OPCODE LENGTH
269             *
270 EF199 0000         CON(5) =THREE        3
          0
271 EF19E 0000         CON(5) =ADD          +          ( * PTR * )
          0
272 EF1A3 0000         CON(5) =DUP          DUP
          0
273 EF1A8 0000         CON(5) =ONE-         1-
          0
274 EF1AD 0000         CON(5) =N@           N@
          0
275 EF1B2 0000         CON(5) =DUP          DUP
          0
276 EF1B7 0000         CON(5) =aOPLEN       OPLEN
          0
277 EF1BC 0000         CON(5) =STORE        !          ( * PTR CNT * )
          0
278             *
279             * GET OPCODE ITSELF
280             *
281 EF1C1 0000         CON(5) =?DUP         ?DUP
          0
282 EF1C6 0000         CON(5) =ZBRNH        IF
          0
283 EF1CB 4600         REL(5) %IF306
          0
284 EF1D0 0000         CON(5) =LIT          5          ( * PTR CNT 5 * )
          0
285 EF1D5 5000         CON(5) 5
          0
286 EF1DA 0000         CON(5) =MIN          MIN
          0
287 EF1DF 0000         CON(5) =DUP          DUP
          0
288 EF1E4 0000         CON(5) =ZERO         0
          0
289 EF1E9 0000         CON(5) =XDO          DO         ( * PTR LIM * )
          0
290 EF1EE       %DO15
```



```
291             *
292             * GET EACH OPCODE SEPARATELY
293             *
294 EF1EE 0000         CON(5) =2DUP         2DUP
          0
295 EF1F3 0000         CON(5) =ADD          +
          0
296 EF1F8 0000         CON(5) =R@           I
          0
297 EF1FD 0000         CON(5) =MINUS        -
          0
298 EF202 0000         CON(5) =ONE-         1-
          0
299 EF207 0000         CON(5) =N@           N@         ( * PTR LIMIT OP * )
          0
300             *
301             * AND STORE IN APPROPRIATE OPx
302             *
303 EF20C 0000         CON(5) =R@           I
          0
304 EF211 0000         CON(5) =ONE+         1+
          0
305 EF216 0000         CON(5) =a>OPx        >OPx
          0
306 EF21B 0000         CON(5) =STORE        !          ( * PTR LIMIT * )
          0
307 EF220 0000         CON(5) =XLOOP        LOOP
          0
308 EF225 9CFF         REL(5) %DO15
          F
309 EF22A 0000         CON(5) =DROP         DROP
          0
310 EF22F       %IF306                      THEN       ( * PTR * )
311             *
312             * GET VARIABLES
313             *
314 EF22F 0000         CON(5) =FIVE+        5+         ( * PTR * )
          0
315 EF234 0000         CON(5) =DUP          DUP
          0
316 EF239 0000         CON(5) =N@           N@
          0
317 EF23E 0000         CON(5) =aFILMK       FILL.MARK
          0
318 EF243 0000         CON(5) =STORE        !
          0
319 EF248 0000         CON(5) =ONE+         1+         ( * PTR * )
          0
320 EF24D 0000         CON(5) =N@           N@
          0
321 EF252 0000         CON(5) =aFILEN       FILL.LENGTH
          0
322 EF257 0000         CON(5) =STORE        !
          0
323 EF25C 0000         CON(5) =SEMI         ;          ( * * )
```

```
          0
   324            ****************************************************************
   325            ***
   326            ***  GETSTR
   327            ***
   328            *** ( ADDR FIR LST TAIL -- ADDR' STR )
   329            *** ( TAKE THE DATA FROM 'ENCLOSE' AND MASSAGE IT AROUND )
   330            *** ( TO CREATE THE NEXT ADDRESS AND THE STRING THAT ENCLOSE )
   331            *** ( GOT. )
   332            ****************************************************************
   333 EF261 0000 =aGTSTR CON(5) =DOCOL         : GETSTR
          0
   334            *
   335            * GET ADDRESS OF BEGINNING OF STRING
   336            *
   337 EF266 0000         CON(5) =LIT           4
          0
   338 EF26B 4000         CON(5) 4
          0
   339 EF270 0000         CON(5) =PICK          PICK
          0
   340 EF275 0000         CON(5) =ADD           +
          0
   341 EF27A 0000         CON(5) =>R            >R          ( * ADDR FIR LST * )
          0
   342            *
   343            * GET LENGTH OF STRING
   344            *
   345 EF27F 0000         CON(5) =OVER          OVER
          0
   346 EF284 0000         CON(5) =MINUS         -
          0
   347 EF289 0000         CON(5) =TWO/          2/
          0
   348 EF28E 0000         CON(5) =>R            >R          ( * ADDR FIR * )
          0
   349            *
   350            * GET NEW ADDRESS
   351            *
   352 EF293 0000         CON(5) =ADD           +
          0
   353            *
   354            * MASSAGE IT AROUND TO CORRECT ORDER
   355            *
   356 EF298 0000         CON(5) =R>            R>
          0
   357 EF29D 0000         CON(5) =R>            R>
          0
   358 EF2A2 0000         CON(5) =ROT           ROT
          0
   359 EF2A7 0000         CON(5) =ROT           ROT         ( * ADDR' ADDR" LEN" *
          0
   360            *
   361            * REMOVE CHARACTER 0 IF IT IS THE LAST CHARACTER
   362            *
```



```
   363 EF2AC 0000         CON(5) =OVER          OVER
          0
   364 EF2B1 0000         CON(5) =CAT           C@
          0
   365 EF2B6 0000         CON(5) =ZEQ           0=
          0
   366 EF2BB 0000         CON(5) =ZBRNH         IF
          0
   367 EF2C0 A000         REL(5) %IF143
          0
   368 EF2C5 0000         CON(5) =ONE-          1-
          0
   369 EF2CA       %IF143                       THEN
   370 EF2CA 0000         CON(5) =SEMI          ;           ( * ADDR' ADDR" LEN" *
          0
   371            ****************************************************************
   372            ***
   373            ***  PARSE
   374            ***
   375            *** ( -- )
   376            *** ( PARSE SOURCE LINE INTO COMPONENT FIELDS )
   377            *** ( AND LOOK THE OPCODE UP IN THE ROM TABLES )
   378            ****************************************************************
   379 EF2CF 0000 =aPARSE CON(5) =DOCOL         : PARSE
          0
   380            *
   381            * APPEND A CHARACTER 0 TO END OF SOURCE LINE
   382            *
   383 EF2D4 0000         CON(5) =aSLINE        SLINE
          0
   384 EF2D9 0000         CON(5) =ZERO          0
          0
   385 EF2DE 0000         CON(5) =CHR$          CHR$
          0
   386 EF2E3 0000         CON(5) =S<&           S<&         ( * STR * )
          0
   387            *
   388            * GET FIRST FIELD
   389            *
   390 EF2E8 0000         CON(5) =DROP          DROP
          0
   391 EF2ED 0000         CON(5) =BL            SPACE
          0
   392 EF2F2 0000         CON(5) =ENCL          ENCLOSE     ( * ADDR FIR LST TAIL *
          0
   393            *
   394            * CHECK FOR FIRST CHARACTER BEING '*'
   395            *
   396 EF2F7 0000         CON(5) =OVER2         2OVER
          0
   397 EF2FC 0000         CON(5) =ADD           +
          0
   398 EF301 0000         CON(5) =CAT           C@
          0
   399 EF306 0000         CON(5) =LIT           42
```

```
          0
400 EF30B A200          CON(5) \*\
          0
401 EF310 0000          CON(5) =EQUAL           =
          0
402 EF315 0000          CON(5) =ZBRNH           IF          ( * ADDR FIR LST TAIL *
          0
403 EF31A B400          REL(5) %IF144
          0
404             *
405             * IT IS, SO TREAT LINE AS A COMMENT
406             *
407 EF31F 0000          CON(5) =aDROP4          4DROP
          0
408 EF324 0000          CON(5) =NULL$           NULL$
          0
409 EF329 0000          CON(5) =2DUP            2DUP
          0
410 EF32E 0000          CON(5) =2DUP            2DUP        ( * STR STR STR * )
          0
411 EF333 0000          CON(5) =aLAB.F          LABEL.FIELD
          0
412 EF338 0000          CON(5) =S!              S!          ( * STR STR * )
          0
413 EF33D 0000          CON(5) =aOPC.F          OPCODE.FIELD
          0
414 EF342 0000          CON(5) =S!              S!          ( * STR * )
          0
415 EF347 0000          CON(5) =aEXP.F          EXPRESSION.FIELD
          0
416 EF34C 0000          CON(5) =S!              S!          ( * * )
          0
417 EF351 0000          CON(5) =LIT             NULL ENTRY
          0
418 EF356 0000          CON(5) =NULENT
          0
419 EF35B D51F          CON(5) =aSTVAR          SET.OP.VARS ( * * )
          E
420 EF360 0000          CON(5) =SEMI            ;           ( * * )
          0
421 EF365       %IF144                          ELSE        ( * ADDR FIR LST TAIL *
422             *
423             * CHECK IF FIRST FIELD BEGINS IN COLUMNS 1 OR 2
424             *
425 EF365 0000          CON(5) =THREE           3
          0
426 EF36A 0000          CON(5) =PICK            PICK
          0
427 EF36F 0000          CON(5) =THREE           3
          0
428 EF374 0000          CON(5) =LT              <
          0
429 EF379 0000          CON(5) =ZBRNH           IF          ( * ADDR FIR LST TAIL *
          0
430 EF37E 7300          REL(5) %IF145
```



```
          0
431             *
432             * IT DOES, SO IT IS A LABEL FIELD
433             *
434 EF383 162F          CON(5) =aGTSTR          GETSTR
          E
435 EF388 0000          CON(5) =LIT             10
          0
436 EF38D A000          CON(5) 10
          0
437 EF392 0000          CON(5) =MIN             MIN         ( * ADDR STR * )
          0
438 EF397 0000          CON(5) =aLAB.F          LABEL.FIELD
          0
439 EF39C 0000          CON(5) =S!              S!          ( * ADDR * )
          0
440             *
441             * GET NEXT FIELD
442             *
443 EF3A1 0000          CON(5) =BL              SPACE
          0
444 EF3A6 0000          CON(5) =ENCL            ENCLOSE     ( * ADDR FIR LST TAIL *
          0
445 EF3AB 0000          CON(5) =BRNCH           ELSE        ( * ADDR FIR LST TAIL *
          0
446 EF3B0 4100          REL(5) %TH145
          0
447 EF3B5       %IF145
448             *
449             * NO LABEL FIELD
450             *
451 EF3B5 0000          CON(5) =NULL$           NULL$       ( * ADDR FIR LST TAIL S
          0
452 EF3BA 0000          CON(5) =aLAB.F          LABEL.FIELD
          0
453 EF3BF 0000          CON(5) =S!              S!          ( * ADDR FIR LST TAIL *
          0
454 EF3C4       %TH145                          THEN        ( * ADDR FIR LST TAIL *
455             *
456             * SAVE OPCODE FIELD
457             *
458 EF3C4 162F          CON(5) =aGTSTR          GETSTR
          E
459 EF3C9 0000          CON(5) =LIT             10
          0
460 EF3CE A000          CON(5) 10
          0
461 EF3D3 0000          CON(5) =MIN             MIN         ( * ADDR STR * )
          0
462 EF3D8 0000          CON(5) =2DUP            2DUP
          0
463 EF3DD 0000          CON(5) =aOPC.F          OPCODE.FIELD
          0
464 EF3E2 0000          CON(5) =S!              S!          ( * ADDR STR * )
          0
```

```
                    *
                    * LOOKUP THE OPCODE IN THE TABLES (ALSO USE HOOKS)
                    *
EF3E7 301F          CON(5) =aOPLUK      OPC.LOOKUP
      E
EF3EC 0000          CON(5) =ZEQ         0=
      0
EF3F1 0000          CON(5) =ZBRNH       IF          ( * ADDR [PTR] * )
      0
EF3F6 E100          REL(5) %IF146
      0
                    *
                    * COULDN'T FIND IT, SUBSTITUTE A NULL OPCODE ENTRY
                    *
EF3FB 0000          CON(5) =LIT         20
      0
EF400 4100          CON(5) 20           'unknown opcode'
      0
EF405 0000          CON(5) =aP2ERR      PASS2 ERROR ( * ADDR * )
      0
EF40A 0000          CON(5) =LIT         NULL ENTRY  ( * ADDR PTR * )
      0
EF40F 0000          CON(5) =NULENT
      0
EF414       %IF146                      THEN
                    *
                    * STORE VARIABLES ASSOCIATED WITH OPCODE
                    *
EF414 D51F          CON(5) =aSTVAR      SET.OP.VARS ( * ADDR * )
      E
                    *
                    * LOOK FOR THE EXPRESSION FIELD
                    *
EF419 0000          CON(5) =BL          SPACE
      0
EF41E 0000          CON(5) =ENCL        ENCLOSE     ( * ADDR FIR LST TAIL *
      0
                    *
                    * COULD THE EXPRESSION FIELD HAVE A QUOTED SPACE ?
                    *
EF423 0000          CON(5) =aOPFLG      OPFLAGS
      0
EF428 0000          CON(5) =AT          @
      0
EF42D 0000          CON(5) =LIT         32
      0
EF432 0200          CON(5) 32
      0
EF437 0000          CON(5) =AND         AND
      0
EF43C 0000          CON(5) =ZBRNH       IF          ( * ADDR FIR LST TAIL *
      0
EF441 8C00          REL(5) %IF148
      0
                    *
```



```
                    * YES, THEN WE HAVE TO GO THROUGH THE EXPRESSION FIELD OURSELVES
                    *
EF446 0000          CON(5) =2DROP       2DROP
      0
EF44B 0000          CON(5) =ADD         +           ( * ADDR * )
      0
EF450 0000          CON(5) =ZERO        0
      0
EF455 0000          CON(5) =OVER        OVER        ( * ADDR F ADDR * )
      0
                    *
                    * LOOP THROUGH EACH CHARACTER UNTIL A SPACE IS REACHED THAT ISN'T
                    *
EF45A       %BG16                       BEGIN       ( * ADDR F ADDR' * )
EF45A 0000          CON(5) =TWO+        2+
      0
EF45F 0000          CON(5) =DUP         DUP
      0
EF464 0000          CON(5) =TWO-        2-
      0
EF469 0000          CON(5) =CAT         C@          ( * ADDR F ADDR' C * )
      0
EF46E 0000          CON(5) =DUP         DUP
      0
EF473 0000          CON(5) =BL          SPACE
      0
EF478 0000          CON(5) =EQUAL       =
      0
EF47D 0000          CON(5) =ZEQ         0=          ( * ADDR F ADDR' C F' *
      0
EF482 0000          CON(5) =LIT         4
      0
EF487 4000          CON(5) 4
      0
EF48C 0000          CON(5) =PICK        PICK
      0
EF491 0000          CON(5) =OR          OR          ( * ADDR F ADDR' C F' *
      0
EF496 0000          CON(5) =OVER        OVER
      0
EF49B 0000          CON(5) =AND         AND         ( * ADDR F ADDR' C F' *
      0
EF4A0 0000          CON(5) =ZBRNH       WHILE       ( * ADDR F ADDR' C * )
      0
EF4A5 7300          REL(5) %RP16
      0
EF4AA 0000          CON(5) =LIT         39
      0
EF4AF 7200          CON(5) \'\
      0
EF4B4 0000          CON(5) =EQUAL       =
      0
EF4B9 0000          CON(5) =ZBRNH       IF          ( * ADDR F ADDR' * )
      0
EF4BE 4100          REL(5) %IF149
```

```
          0
    532           *
    533           * WE'VE REACHED A QUOTE ('), SO REVERSE OUR FLAG
    534           *
    535 EF4C3 0000        CON(5) =SWAP          SWAP
          0
    536 EF4C8 0000        CON(5) =NOT           NOT
          0
    537 EF4CD 0000        CON(5) =SWAP          SWAP        ( * ADDR F ADDR' * )
          0
    538 EF4D2      %IF149                       THEN        ( * ADDR F ADDR' * )
    539 EF4D2 0000        CON(5) =BRNCH         REPEAT      ( * ADDR F ADDR' C * )
          0
    540 EF4D7 38FF        REL(5) %BG16
          F
    541           *
    542           * EXPRESSION FIELD NOW MARKED, LET'S MAKE IT A STRING
    543           *
    544 EF4DC      %RP16
    545 EF4DC 0000        CON(5) =DROP          DROP
          0
    546 EF4E1 0000        CON(5) =SWAP          SWAP
          0
    547 EF4E6 0000        CON(5) =DROP          DROP        ( * ADDR ADDR' * )
          0
    548 EF4EB 0000        CON(5) =OVER          OVER
          0
    549 EF4F0 0000        CON(5) =MINUS         -
          0
    550 EF4F5 0000        CON(5) =TWO-          2-
          0
    551 EF4FA 0000        CON(5) =TWO/          2/          ( * STR * )
          0
    552 EF4FF 0000        CON(5) =BRNCH         ELSE        ( * ADDR FIR LST TAIL *
          0
    553 EF504 4100        REL(5) %TH148
          0
    554 EF509      %IF148                       ELSE        ( * ADDR FIR LST TAIL *
    555           *
    556           * EXPRESSION CANNOT HAVE QUOTED SPACES
    557           *
    558 EF509 162F        CON(5) =aGTSTR        GETSTR
          E
    559 EF50E 0000        CON(5) =ROT           ROT
          0
    560 EF513 0000        CON(5) =DROP          DROP        ( * STR * )
          0
    561 EF518      %TH148                       THEN
    562           *
    563           * ENSURE STRING UNDER 51 CHARACTERS
    564
    565 EF518 0000        CON(5) =DUP           DUP         ( * STR LEN * )
          0
    566 EF51D 0000        CON(5) =LIT           50
          0
```



```
    567 EF522 2300        CON(5) 50
          0
    568 EF527 0000        CON(5) =GT            >           ( * STR F * )
          0
    569 EF52C 0000        CON(5) =ZBRNH         IF          ( * STR * )
          0
    570 EF531 3200        REL(5) %IF138
          0
    571           *
    572           * EXPRESSION FIELD TOO LONG
    573           *
    574 EF536 0000        CON(5) =LIT           40
          0
    575 EF53B 8200        CON(5) 40             'excess characters in expression'
          0
    576 EF540 0000        CON(5) =aP2ERR        PASS2 ERROR ( * STR * )
          0
    577 EF545 0000        CON(5) =DROP          DROP        ( * ADDR * )
          0
    578 EF54A 0000        CON(5) =LIT           50
          0
    579 EF54F 2300        CON(5) 50                         ( * STR * )
          0
    580 EF554      %IF138                       THEN        ( * STR * )
    581           *
    582           * SAVE EXPRESSION FIELD IN VARIABLE
    583           *
    584 EF554 0000        CON(5) =aEXP.F        EXPRESSION.FIELD
          0
    585 EF559 0000        CON(5) =S!            S!          ( * * )
          0
    586 EF55E 0000        CON(5) =SEMI          ;           ( * * )
          0
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER

```
                        STITLE FIT STATS
           zSIZE   EQU     (*)-zTHIS
           zLEFT   EQU     (zNEXT)-*
EF563                   BSS     zLEFT
EF600                   END
```


%BG15	Abs	978888	#EEFC8	-	97	185						
%BG16	Abs	980058	#EF45A	-	510	540						
%D014	Abs	978718	#EEF1E	-	34	49						
%D015	Abs	979438	#EF1EE	-	290	308						
%IF138	Abs	980308	#EF554	-	580	570						
%IF139	Abs	979038	#EF05E	-	143	137						
%IF140	Abs	979118	#EF0AE	-	166	157						
%IF141	Abs	979153	#EF0D1	-	180	173						
%IF142	Abs	979198	#EF0FE	-	198	191						
%IF143	Abs	979658	#EF2CA	-	369	367						
%IF144	Abs	979813	#EF365	-	421	403						
%IF145	Abs	979893	#EF3B5	-	447	430						
%IF146	Abs	979988	#EF414	-	480	471						
%IF148	Abs	980233	#EF509	-	554	499						
%IF149	Abs	980178	#EF4D2	-	538	531						
%IF150	Abs	979263	#EF13F	-	230	224						
%IF303	Abs	979268	#EF144	-	235	218						
%IF306	Abs	979503	#EF22F	-	310	283						
%RP16	Abs	980188	#EF4DC	-	544	526						
%TH139	Abs	979158	#EF0D6	-	183	142						
%TH140	Abs	979158	#EF0D6	-	182	165						
%TH145	Abs	979908	#EF3C4	-	454	446						
%TH148	Abs	980248	#EF518	-	561	553						
2DROP	Ext			-	54	503						
2DUP	Ext			-	236	294	409	410	462			
>R	Ext			-	35	53	341	348				
?DUP	Ext			-	216	281						
ADD	Ext			-	47	78	116	127	151	262	271	295
					340	352	397	504				
AND	Ext			-	39	155	497	524				
AT	Ext			-	79	102	115	148	196	215	494	
BL	Ext			-	391	443	488	516				
BRNCH	Ext			-	141	164	445	539	552			
C@+	Ext			-	36							
CAT	Ext			-	152	254	264	364	398	514		
CHR$	Ext			-	385							
DOCOL	Ext			-	22	69	88	209	249	333	379	
DROP	Ext			-	309	390	545	547	560	577		
DUP	Ext			-	103	120	128	189	250	263	272	275
					287	315	512	515	565			
ENCL	Ext			-	392	444	489					
EQUAL	Ext			-	401	517	529					
EXEC	Ext			-	222							
FIVE+	Ext			-	314							
GT	Ext			-	568							
LIT	Ext			-	27	37	44	59	73	76	111	149
					153	213	260	284	337	399	417	435
					459	475	478	495	519	527	566	574
					578							
LT	Ext			-	428							
MIN	Ext			-	286	437	461					
MINUS	Ext			-	297	346	549					
MOD	Ext			-	61							
MULT	Ext			-	46	75	113					
N@	Ext			-	130	274	299	316	320			

NOT	Ext		-	536							
NULENT	Ext		-	418	479						
NULL$	Ext		-	408	451						
ONE	Ext		-	94	177						
ONE+	Ext		-	304	319						
ONE-	Ext		-	110	129	273	298	368			
OPTABL	Ext		-	77							
OR	Ext		-	522							
OVER	Ext		-	345	363	506	523	548			
OVER2	Ext		-	170	396						
PICK	Ext		-	339	426	521					
PSTOR	Ext		-	179							
R>	Ext		-	40	55	356	357				
R@	Ext		-	296	303						
ROT	Ext		-	358	359	559					
S!	Ext		-	412	414	416	439	453	464	585	
S<	Ext		-	171							
S<&	Ext		-	386							
S=	Ext		-	135							
SEMI	Ext		-	62	80	199	229	240	323	370	420
				586							
STORE	Ext		-	93	96	106	122	256	266	277	306
				318	322						
SWAP	Ext		-	197	535	537	546				
THREE	Ext		-	126	270	425	427				
TWO*	Ext		-	104							
TWO+	Ext		-	511							
TWO-	Ext		-	513	550						
TWO/	Ext		-	347	551						
XDO	Ext		-	30	289						
XLOOP	Ext		-	48	307						
ZBRNH	Ext		-	136	156	172	184	190	217	223	282
				366	402	429	470	498	525	530	569
ZEQ	Ext		-	365	469	518					
ZERO	Ext		-	26	29	162	181	288	384	505	
a>OPx	Ext		-	305							
aDROP4	Ext		-	138	161	407					
aDUP4	Ext		-	134							
aEXP.F	Ext		-	415	584						
aFILEN	Ext		-	321							
aFILMK	Ext		-	317							
=aGTSTR	Abs	979553 #EF261	-	333	434	458	558				
=aHASH	Abs	978688 #EEF00	-	22	237						
=aJUMP	Abs	978813 #EEF7D	-	69	238						
aLAB.F	Ext		-	411	438	452					
aNEG1	Ext		-	139	140	163	228				
aOPC.F	Ext		-	234	413	463					
aOPCPO	Ext		-	95	101	105	178				
aOPCPT	Ext		-	121	147	195					
aOPCTO	Ext		-	92	114						
aOPFLG	Ext		-	265	493						
aOPLEN	Ext		-	276							
=aOPLUK	Abs	979203 #EF103	-	209	468						
aOPTYP	Ext		-	255							
aP2ERR	Ext		-	477	576						

=aPARSE	Abs	979663 #EF2CF	-	379			
aSLINE	Ext		-	383			
=aSRCH	Abs	978858 #EEFAA	-	88	239		
=aSTVAR	Abs	979293 #EF15D	-	249	419	484	
zLEFT	Abs	157 #0009D	-	589	590		
zNEXT	Abs	980480 #EF600	-	5	589		
zSIZE	Abs	1635 #00663	-	588			
zTHIS	Abs	978688 #EEF00	-	4	588		

Input Parameters

Source file name is GN&AS7

Listing file name is GN/AS7::65

Object file name is GN%AS7::65

```
                                    111111
                          0123456789012345
```

Initial flag settings are

Errors

None

Saturn Assembler News

```
 1                      TITLE  AS8:_FORTH_ASSEMBLER_MAIN
 2                      RDSYMB MR%GTO
 3 EF600                ABS    #EF600
 4                zTHIS  EQU    *
 5                zNEXT  EQU    #EFF00
 6
 7                ******************************************************
 8                ***
 9                *** MR&AS8     <840504.1037>
10                ***
11                ***
12                ***  FORTH ASSEMBLER
13                ***  ONLY THE LAST WORD, 'ASSEMBLE' HAS A HEADER
14                ***
15                ******************************************************
16                ******************************************************************
17                ***
18                ***  MONITOR.ATTN
19                ***
20                *** ( -- )
21                *** ( WATCH THE INPUT BUFFER FOR THE ATTN KEY )
22                ******************************************************************
23 EF600 0000 =aMONAT CON(5) =DOCOL          : MONITOR.ATTN
          0
24                *
25                * while <KEYS HIT> do
26                *
27 EF605           %BG17                     BEGIN       ( * * )
28 EF605 0000             CON(5) =?TERM      ?TERMINAL   ( * F * )
          0
29 EF60A 0000             CON(5) =ZBRNH      WHILE       ( * * )
          0
30 EF60F BC00             REL(5) %RP17
          0
31                *
32                * IS KEY THE ATTN KEY?
33                *
34 EF614 0000             CON(5) =KEY        KEY         ( * NUM * )
          0
35 EF619 0000             CON(5) =LIT        14          ( * NUM NUM' * )
          0
36 EF61E E000             CON(5) 14
          0
37 EF623 0000             CON(5) =EQUAL      =           ( * F * )
          0
38 EF628 0000             CON(5) =ZBRNH      IF          ( * * )
          0
39 EF62D 3A00             REL(5) %IF150
          0
40                *
41                * YES.  DISPLAY ABORT MESSAGE
42                *
43 EF632 0000             CON(5) =CR         CR          ( * * )
          0
44 EF637 0000             CON(5) =PDOTQ      ." ABORT [Y/N] ?"
```



```
          0
45 EF63C D0               CON(2) 13
46 EF63E 1424             NIBASC \ABORT [Y\
         F425
         4502
         B595
47 EF64E F2E4             NIBASC \/N] ?\
         D502
         F3
48                *
49                * while <KEYS HIT> do
50                *
51 EF658           %BG19                     BEGIN       ( * * )
52 EF658 0000             CON(5) =?TERM      ?TERMINAL   ( * F * )
          0
53 EF65D 0000             CON(5) =ZBRNH      WHILE       ( * * )
          0
54 EF662 9100             REL(5) %RP19
          0
55                *
56                * DROP KEY
57                *
58 EF667 0000             CON(5) =KEY        KEY         ( * NUM * )
          0
59 EF66C 0000             CON(5) =DROP       DROP        ( * * )
          0
60 EF671 0000             CON(5) =BRNCH      REPEAT      ( * * )
          0
61 EF676 2EFF             REL(5) %BG19
          F
62                *
63                * end while
64                *
65 EF67B           %RP19
66                *
67                * ASK FOR A KEY
68                *
69 EF67B 0000             CON(5) =KEY        KEY         ( * NUM * )
          0
70 EF680 0000             CON(5) =DUP        DUP
          0
71 EF685 0000             CON(5) =EMIT       EMIT        ( * NUM * )
          0
72 EF68A 0000             CON(5) =LIT        89
          0
73 EF68F 9500             CON(5) 89
          0
74 EF694 0000             CON(5) =DUP        DUP         ( * NUM 89 89 * )
          0
75 EF699 0000             CON(5) =ROT        ROT         ( * 89 89 NUM * )
          0
76 EF69E 0000             CON(5) =AND        AND         ( * 89 NUM * )
          0
77 EF6A3 0000             CON(5) =EQUAL      =           ( * F * )
          0
```

```
 78 EF6A8 0000         CON(5) =ZBRNH          IF          ( * * )
          0
 79 EF6AD 4100         REL(5) %IF152
          0
 80             *
 81             * USER HIT Y OR y.  SHUT DOWN
 82             *
 83 EF6B2 0000         CON(5) =LIT            56
          0
 84 EF6B7 8300         CON(5) 56              'assembler aborted'
          0
 85 EF6BC 0000         CON(5) =aSHUTD         SHUTDOWN
          0
 86             *
 87             * USER HIT A DIFFERENT KEY.
 88             *
 89 EF6C1       %IF152                        THEN        ( * * )
 90             *
 91             * DISPLAY OF PROGRESS MUFFED.  RESET DOT COUNT.
 92             *
 93 EF6C1 0000         CON(5) =ZERO           0
          0
 94 EF6C6 0000         CON(5) =aLINEC         LINECOUNT
          0
 95 EF6CB 0000         CON(5) =STORE          !           ( * * )
          0
 96 EF6D0       %IF150                        THEN        ( * * )
 97 EF6D0 0000         CON(5) =BRNCH          REPEAT
          0
 98 EF6D5 03FF         REL(5) %BG17
          F
 99             *
100             * end while
101             *
102 EF6DA       %RP17
103 EF6DA 0000         CON(5) =SEMI           ;
          0
104             ****************************************************************
105             ***
106             ***  SHOW.PROGRESS
107             ***
108             *** ( -- )
109             *** ( INDICATE ON THE DISPLAY THAT ACTIONS OCCURS )
110             ****************************************************************
111 EF6DF 0000 =aSHOWP CON(5) =DOCOL          : SHOW.PROGRESS
          0
112             *
113             * SHOULD WE DISPLAY A NEW LINE?
114             *
115 EF6E4 0000         CON(5) =aLINEC         LINECNT
          0
116 EF6E9 0000         CON(5) =AT             @
          0
117 EF6EE 0000         CON(5) =LIT            14
          0
```



```
118 EF6F3 E000         CON(5) 14
          0
119 EF6F8 0000         CON(5) =MOD            MOD
          0
120 EF6FD 0000         CON(5) =ZEQ            0=
          0
121 EF702 0000         CON(5) =ZBRNH          IF          ( * * )
          0
122 EF707 B300         REL(5) %IF153
          0
123             *
124             * YES, NUMBER OF DOTS FILLS SCREEN
125             *
126 EF70C 0000         CON(5) =CR             CR
          0
127             *
128             * DISPLAY PASS
129             *
130 EF711 0000         CON(5) =QUOTC          "
          0
131 EF716 50           CON(2) 5
132 EF718 50           CON(2) 5
133 EF71A 0514         NIBASC \PASS \
          3535
          02
134 EF724 0000         CON(5) =TYPE           TYPE        ( * * )
          0
135 EF729 0000         CON(5) =aPASS          PASS
          0
136 EF72E 0000         CON(5) =AT             @
          0
137 EF733 0000         CON(5) =aNUM#1         #1
          0
138 EF738 0000         CON(5) =TYPE           TYPE        ( * * )
          0
139 EF73D 0000         CON(5) =SPACE          SPACE       ( * * )
          0
140             *
141             * ADD ANOTHER DOT TO SCREEN
142             *
143 EF742       %IF153                        THEN
144 EF742 0000         CON(5) =LIT            46
          0
145 EF747 E200         CON(5) 46
          0
146 EF74C 0000         CON(5) =EMIT           EMIT        ( * * )
          0
147 EF751 0000         CON(5) =ONE            1
          0
148 EF756 0000         CON(5) =aLINEC         LINECNT
          0
149 EF75B 0000         CON(5) =PSTOR          +!
          0
150 EF760 0000         CON(5) =SEMI           ;           ( * * )
          0
```

```
                ****************************************************************
                ***
                ***  NOTHING
                ***
                ****************************************************************
EF765 00000 =aBLANK CON(5) =DOCOL          : NOTHING
EF76A 00000         CON(5) =SEMI           ;
                *** ( -- )
                *** ( DO NOTHING AT ALL, ACTUALLY JUST A PLACE HOLDER )
                *** ( FOR THE TABLE THAT FOLLOWS )
                ****************************************************************
                ***
                ***  TABLE
                ***
                *** ( -- )
                *** ( THIS IS NEVER EXECUTED, BUT DOES PROVIDE )
                *** ( A TABLE OF ADDRESS FOR PROCESS )
                *** ( 00-04 )
                ****************************************************************
EF76F           =aTABLE                    : TABLE
EF76F 567FE         CON(5) =aBLANK         NOTHING
EF774 00000         CON(5) =aREGTE         REGTEST
EF779 00000         CON(5) =aREGAR         REGARITH
EF77E 00000         CON(5) =aREGLO         REGLOGIC
EF783 00000         CON(5) =aBRANC         BRANCHES
                *** ( 05-09 )
EF788 567FE         CON(5) =aBLANK         NOTHING
EF78D 567FE         CON(5) =aBLANK         NOTHING
EF792 00000         CON(5) =aRTNYS         RTNYES
EF797 00000         CON(5) =aPTRTE         PTRTEST
EF79C 00000         CON(5) =aSTATT         STATTEST
                *** ( 10-14 )
EF7A1 00000         CON(5) =aSETPT         SETPTR
EF7A6 00000         CON(5) =aSETST         SETSTAT
EF7AB 00000         CON(5) =aDPARI         DPARITH
EF7B0 00000         CON(5) =aDATAT         DATATRANS
EF7B5 00000         CON(5) =aNIBHX         NIBHEX
                *** ( 15-19 )
```



```
EF7BA 00000         CON(5) =aLCHEX         LCHEX
EF7BF 00000         CON(5) =Dx=HX          Dx=HEX
EF7C4 00000         CON(5) =aNIBFF         NIBASC
EF7C9 00000         CON(5) =aLCASC         LCASC
EF7CE 00000         CON(5) =aBSS           BSS
                *** ( 20-24 )
EF7D3 00000         CON(5) =aEJECT         EJECT
EF7D8 00000         CON(5) =aEND           END
EF7DD 00000         CON(5) =aEQU           EQU
EF7E2 00000         CON(5) =aLIST          LIST
EF7E7 00000         CON(5) =aTITLE         TITLE
                *** ( 25-29 )
EF7EC 00000         CON(5) =aSTITL         STITLE
EF7F1 00000         CON(5) =aWORDN         WORDN
EF7F6 00000         CON(5) =aWORDI         WORDI
EF7FB 00000         CON(5) =aLEX           LEX
EF800 00000         CON(5) =aIDN           ID
                *** ( 30-34 )
EF805 00000         CON(5) =aMSG           MSG
EF80A 00000         CON(5) =aPOLL          POLL
EF80F 00000         CON(5) =aENTRY         ENTRY
EF814 00000         CON(5) =aCHAR          CHAR
EF819 00000         CON(5) =aKEYP          KEYP
                *** ( 35-39 )
EF81E 00000         CON(5) =aTOKNP         TOKENP
EF823 00000         CON(5) =aBIN           BIN
EF828 00000         CON(5) =aFORTH         FORTHN
EF82D 00000         CON(5) =aCHAIN         CHAIN
EF832 00000         CON(5) =aENDTX         END.TEXT
                ****************************************************************
```

```
                ***
                ***  PROCESS
                ***
                *** ( -- )
                *** ( GO TO THE PROPER OPCODE ALGORITHM )
                *** ( IF AN ALGORITHM RETURNS 'NUM' IT IS PASS 1 )
                ********************************************************************
EF837 0000 =aPROCS CON(5) =DOCOL          : PROCESS
           0
                *
                * LOOK UP WHICH PROCESS ROUTINE TO CALL
                *
EF83C 0000         CON(5) =LIT            TABLE       ( * ADDR * )
           0
EF841 F67F         CON(5) =aTABLE
           E
EF846 0000         CON(5) =aOPTYP         OPTYPE
           0
EF84B 0000         CON(5) =AT             @           ( * ADDR NUM * )
           0
EF850 0000         CON(5) =DUP            DUP         ( * ADDR NUM NUM * )
           0
EF855 0000         CON(5) =LIT            39
           0
EF85A 7200         CON(5) 39
           0
EF85F 0000         CON(5) =GT             >           ( * ADDR NUM F * )
           0
EF864 0000         CON(5) =ZBRNH          IF          ( * ADDR NUM * )
           0
EF869 3200         REL(5) %IF304
           0
                *
                * THE HOOKS HAVE PROVIDED A PROCESS ROUTINE WITH NUMBER > 39.
                * SINCE THIS COULDN'T HAPPEN INTERNALLY, WE'LL LET THEM
                * HANDLE IT.
                *
EF86E 0000         CON(5) =SWAP           SWAP        ( * NUM ADDR * )
           0
EF873 0000         CON(5) =DROP           DROP        ( * NUM * )
           0
EF878 0000         CON(5) =LIT            PROCESS RT  ( * NUM PTR * )
           0
EF87D E7CF         CON(5) =oPRORT
           2
EF882 0000         CON(5) =BRNCH          ELSE
           0
EF887 9100         REL(5) %TH304
           0
                *
                * PULL OUR PROCESS ROUTINE FROM TABLE
                *
EF88C       %IF304
EF88C 0000         CON(5) =LIT            5
           0
```



```
EF891 5000         CON(5) 5
           0
EF896 0000         CON(5) =MULT           *
           0
EF89B 0000         CON(5) =ADD            +           ( * PTR * )
           0
                *
                * EXECUTE PROCESS ROUTINE
                *
EF8A0       %TH304                        THEN        ( * [NUM] PTR * )
EF8A0 0000         CON(5) =AT             @
           0
EF8A5 0000         CON(5) =EXEC           EXECUTE     ( * [NUM] * )
           0
                *
                * IF PASS 1, IT SHOULD RETURN A LENGTH
                *
EF8AA 0000         CON(5) =aPASS          PASS
           0
EF8AF 0000         CON(5) =AT             @
           0
EF8B4 0000         CON(5) =ONE            1
           0
EF8B9 0000         CON(5) =EQUAL          =
           0
EF8BE 0000         CON(5) =ZBRNH          IF          ( * [NUM] * )
           0
EF8C3 F000         REL(5) %IF154
           0
                *
                * WE'LL STORE THIS LENGTH IN VARIABLE OPLEN
                *
EF8C8 0000         CON(5) =aOPLEN         OPLEN
           0
EF8CD 0000         CON(5) =STORE          !           ( * * )
           0
                *
                * ALL DONE
                *
EF8D2       %IF154                        THEN
EF8D2 0000         CON(5) =SEMI           ;           ( * * )
           0
                ********************************************************************
                ***
                ***  SPIT
                ***
                *** ( -- )
                *** ( SPIT PROCESSED INFORMATION TO OBJECT & LISTING )
                ********************************************************************
EF8D7 0000 =aSPIT  CON(5) =DOCOL          : SPIT
           0
                *
                * CHECK FOR TEST INSTRUCTION/RTNYES MATCH
                *
EF8DC 0000         CON(5) =aLSTRQ         LAST.REQ
```

```
          0
  296 EF8E1 0000        CON(5) =AT            @
          0
  297 EF8E6 0000        CON(5) =ZBRNH         IF          ( * * )
          0
  298 EF8EB 7300        REL(5) %IF155
          0
  299 EF8F0 0000        CON(5) =aISGYS        IS.GOYES
          0
  300 EF8F5 0000        CON(5) =ZEQ           0=          ( * F * )
          0
  301           *
  302           * BUT ONLY IF THE LINE GENERATES CODE
  303           *
  304 EF8FA 0000        CON(5) =aOPLEN        OPLEN
          0
  305 EF8FF 0000        CON(5) =AT            @
          0
  306 EF904 0000        CON(5) =F.AND         F.AND       ( * F * )
          0
  307 EF909 0000        CON(5) =ZBRNH         IF
          0
  308 EF90E 4100        REL(5) %IF156
          0
  309 EF913 0000        CON(5) =LIT           21
          0
  310           *
  311           * REPORT ERROR
  312           *
  313 EF918 5100        CON(5) 21             'requires previous GOYES/RTNYES'
          0
  314 EF91D 0000        CON(5) =aERROR        ERROR
          0
  315 EF922      %IF156                       THEN
  316 EF922      %IF155                       THEN        ( * * )
  317           *
  318           * OUTPUT OBJECT CODE & LISTING LINE
  319           *
  320 EF922 0000        CON(5) =aOUTOB        OUT.OBJ
          0
  321 EF927 0000        CON(5) =aOUTLS        OUT.LIST
          0
  322 EF92C 0000        CON(5) =SEMI          ;           ( * * )
          0
  323           ****************************************************************
  324           ***
  325           ***  LC+
  326           ***
  327           *** ( -- )
  328           *** ( SET LAST.REQ AND UPDATE LC )
  329           ****************************************************************
  330 EF931 0000 =aLC+  CON(5) =DOCOL         : LC+
          0
  331 EF936 0000        CON(5) =aOPLEN        OPLEN
          0
```



```
  332 EF93B 0000        CON(5) =AT            @
          0
  333 EF940 0000        CON(5) =ZBRNH         IF          ( * * )
          0
  334 EF945 8200        REL(5) %IF158
          0
  335           *
  336           * THIS LINE DID GENERATE CODE.  SET VARIABLE LAST.REQ
  337           *
  338 EF94A 0000        CON(5) =aRQYES        REQ.YES
          0
  339 EF94F 0000        CON(5) =aLSTRQ        LAST.REQ
          0
  340 EF954 0000        CON(5) =STORE         !           ( * * )
          0
  341           *
  342           * UPDATE VARIABLE LC
  343           *
  344 EF959 0000        CON(5) =aOPLEN        OPLEN
          0
  345 EF95E 0000        CON(5) =AT            @
          0
  346 EF963 0000        CON(5) =aLC           LC
          0
  347 EF968 0000        CON(5) =PSTOR         +!
          0
  348           *
  349           * ALL DONE
  350           *
  351 EF96D      %IF158                       THEN
  352 EF96D 0000        CON(5) =SEMI          ;           ( * * )
          0
  353           ****************************************************************
  354           ***
  355           ***  LABELS
  356           ***
  357           *** ( -- )
  358           *** ( PUT LABELS INTO THE SYMBOL TABLE )
  359           ****************************************************************
  360 EF972 0000 =aLABLS CON(5) =DOCOL        : LABELS
          0
  361           *
  362           * DOES LABEL EXIST?
  363           *
  364 EF977 0000        CON(5) =aLAB.F        LABEL.FIELD ( * STR * )
          0
  365 EF97C 0000        CON(5) =SWAP          SWAP        ( * LEN ADDR * )
          0
  366 EF981 0000        CON(5) =DROP          DROP        ( * LEN * )
          0
  367 EF986 0000        CON(5) =ZBRNH         IF          ( * * )
          0
  368 EF98B 7800        REL(5) %IF160
          0
  369           *
```

```
370           * YES.  GET VALUE FOR LABEL AND LABEL FIELD
371           *
372 EF990 0000        CON(5) =aLC           LC           ( * PTR * )
          0
373 EF995 0000        CON(5) =AT            @            ( * NUM * )
          0
374 EF99A 0000        CON(5) =aLAB.F        LABEL.FIELD  ( * NUM STR * )
          0
375 EF99F 0000        CON(5) =a?PAS1        ?PASS=1      ( * NUM STR F * )
          0
376 EF9A4 0000        CON(5) =ZBRNH         IF           ( * NUM STR * )
          0
377 EF9A9 F000        REL(5) %IF161
          0
378           *
379           * PUT ONTO SYMBOL TABLE IF PASS 1
380           *
381 EF9AE 0000        CON(5) =a>SYT         >SYT         ( * * )
          0
382 EF9B3 0000        CON(5) =SEMI          ;            ( * * )
          0
383           *
384           * CHECK LABEL FOR VALIDITY IF PASS 2
385           *
386 EF9B8       %IF161                                    ( * NUM STR * )
387 EF9B8 0000        CON(5) =2DUP          2DUP         ( * NUM STR STR * )
          0
388 EF9BD 0000        CON(5) =aLABOK        LABEL.OK?    ( * NUM STR F * )
          0
389 EF9C2 0000        CON(5) =ZBRNH         IF           ( * NUM STR * )
          0
390 EF9C7 2300        REL(5) %IF162
          0
391           *
392           * VALID LABEL, BUT HAS IT'S VALUE CHANGED FROM THE FIRST PASS ?
393           *
394 EF9CC 0000        CON(5) =aSYT>         SYT>         ( * NUM NUM' * )
          0
395 EF9D1 0000        CON(5) =EQUAL         =            ( * F * )
          0
396 EF9D6 0000        CON(5) =ZEQ           0=           ( * F * )
          0
397 EF9DB 0000        CON(5) =ZBRNH         IF           ( * * )
          0
398 EF9E0 2300        REL(5) %IF160
          0
399           *
400           * YES.  EITHER CODE SHIFTED (IMPOSSIBLE!  UNLESS HOOKS GENERATED
401           * DIFFERENT LENGTH IN PASS2 THAN REPORTED IN PASS 1!)  OR THERE W
402           * ANOTHER LABEL BY THE SAME NAME.
403           *
404 EF9E5 0000        CON(5) =LIT           41           ( * NUM * )
          0
405 EF9EA 9200        CON(5) 41             'duplicate label'
          0
```



```
406 EF9EF 0000        CON(5) =aERROR        ERROR        ( * * )
          0
407 EF9F4 0000        CON(5) =SEMI          ;            ( * * )
          0
408           *
409           * INVALID LABEL ON PASS 2
410           *
411 EF9F9       %IF162                      ELSE         ( * NUM STR * )
412 EF9F9 0000        CON(5) =2DROP         2DROP        ( * NUM * )
          0
413 EF9FE 0000        CON(5) =DROP          DROP         ( * * )
          0
414 EFA03 0000        CON(5) =LIT           22           ( * NUM * )
          0
415 EFA08 6100        CON(5) '22            'illegal label'
          0
416 EFA0D 0000        CON(5) =aERROR        ERROR        ( * * )
          0
417           *
418           * ALL DONE
419           *
420 EFA12       %IF160                      THEN         ( * * )
421 EFA12 0000        CON(5) =SEMI          ;            ( * * )
          0
422           ****************************************************************
423           ***
424           ***  DO.LINE
425           ***
426           *** ( -- )
427           *** ( TAKE A SOURCE LINE AND PRODUCE CODE )
428           ****************************************************************
429 EFA17 0000 =aDOLIN CON(5) =DOCOL         : DO.LINE
          0
430           *
431           * MONITOR ATTENTION KEY & SHOW NEW LINE PROGRESS
432           *
433 EFA1C 006F        CON(5) =aMONAT        MONIT.ATTN   ( * * )
          E
434 EFA21 FD6F        CON(5) =aSHOWP        PROGRESS     ( * * )
          E
435           *
436           * PARSE SOURCE LINE
437           *
438 EFA26 0000        CON(5) =aPARSE        PARSE        ( * * )
          0
439           *
440           * CHECK LABEL FIELD, UNLESS WE'RE PROCESSING AN 'EQU' STATEMENT.
441           * IF SO, THE PSEUDO-OP WILL PROCESS THE LABEL ITSELF.
442           *
443 EFA2B 0000        CON(5) =aOPTYP        OPTYPE
          0
444 EFA30 0000        CON(5) =AT            @            ( * NUM * )
          0
445 EFA35 0000        CON(5) =LIT           22
          0
```

```
446 EFA3A 6100          CON(5) 22
          0
447 EFA3F 0000          CON(5) =EQUAL           =           ( * F * )
          0
448 EFA44 0000          CON(5) =ZEQ             0=          ( * F * )
          0
449 EFA49 0000          CON(5) =ZBRNH           IF
          0
450 EFA4E A000          REL(5) %1F167
          0
451 EFA53 279F          CON(5) =aLABLS          LABELS
          E
452 EFA58       %IF167                          THEN
453             *
454             * PROCESS LINE
455             *
456 EFA58 0000          CON(5) =a?PAS1          ?PASS=1     ( * F * )
          0
457 EFA5D 0000          CON(5) =ZBRNH           IF          ( * * )
          0
458 EFA62 6400          REL(5) %IF311
          0
459             *
460             * PROCESSING FOR PASS 1 HERE
461             *
462 EFA67 0000          CON(5) =aLENST          LENGTH.SET
          0
463 EFA6C 0000          CON(5) =ZEQ             0=
          0
464 EFA71 0000          CON(5) =ZBRNH           IF          ( * * )
          0
465 EFA76 A000          REL(5) %IF157
          0
466 EFA7B 738F          CON(5) =aPROCS          PROCESS     ( * * )
          E
467             *
468             * CHECK THAT THE FILETYPE HAS BEEN DECLARED
469             *
470 EFA80       %IF157                          THEN        ( * * )
471 EFA80 0000          CON(5) =aFLTYP          FILETYPE
          0
472 EFA85 0000          CON(5) =AT              @           ( * NUM * )
          0
473 EFA8A 0000          CON(5) =ZEQ             0=          ( * F * )
          0
474 EFA8F 0000          CON(5) =ZBRNH           IF          ( * * )
          0
475 EFA94 E100          REL(5) %TH311
          0
476 EFA99 0000          CON(5) =LIT             51
          0
477 EFA9E 3300          CON(5) 51               'missing file type'
          0
478 EFAA3 0000          CON(5) =aSHUTD          SHUTDOWN    ( * * )
          0
```



```
479             *
480             * PROCESSING FOR PASS 2 HERE
481             *
482 EFAA8       %IF311                          ELSE        ( * * )
483 EFAA8 738F          CON(5) =aPROCS          PROCESS     ( * * )
          E
484 EFAAD 7D8F          CON(5) =aSPIT           SPIT        ( * * )
          E
485             *
486             * UPDATE LOCATION COUNTER & RESET ERROR-THIS-LINE FLAG
487             *
488 EFAB2       %TH311                          THEN        ( * * )
489 EFAB2 139F          CON(5) =aLC+            LC+         ( * * )
          E
490 EFAB7 0000          CON(5) =LIT             ~8
          0
491 EFABC 7FFF          CON(5) #FFFF7
          F
492 EFAC1 0000          CON(5) =aTESTF          TEST.FLAG   ( * F * )
          0
493 EFAC6 0000          CON(5) =aOCCUR          OCCUR
          0
494 EFACB 0000          CON(5) =STORE           !           ( * * )
          0
495 EFAD0 0000          CON(5) =SEMI            ;           ( * * )
          0
496             ***************************************************************
497             ***
498             ***  PASS.INIT
499             ***
500             *** ( -- )
501             *** ( INITIALIZE APPROPRIATE VARIABLES AT )
502             *** ( BEGINNING OF PASS )
503             ***************************************************************
504 EFAD5 0000 =aPASIN CON(5) =DOCOL           : PASS.INIT
          0
505 EFADA 0000          CON(5) =a?PAS1          ?PASS=1
          0
506 EFADF 0000          CON(5) =ZBRNH           IF          ( * * )
          0
507 EFAE4 7300          REL(5) %IF163
          0
508             *
509             * LOW TOKEN COUNT TO HIGHEST POSSIBLE VALUE
510             *
511 EFAE9 0000          CON(5) =LIT             255
          0
512 EFAEE FF00          CON(5) 255
          0
513 EFAF3 0000          CON(5) =aLOTOK          LOTOK
          0
514 EFAF8 0000          CON(5) =STORE           !           ( * * )
          0
515             *
516             * HIGH TOKEN COUNT TO LOWEST POSSIBLE VALUE
```

```
                *
EFAFD 0000          CON(5) =LIT          -256
          0
EFB02 00FF          CON(5) 0-256
          F
EFB07 0000          CON(5) =aHITOK       HITOK
          0
EFB0C 0000          CON(5) =STORE        !            ( * * )
          0
EFB11 0000          CON(5) =BRNCH        ELSE
          0
EFB16 4600          REL(5) %TH163
          0
EFB1B              %IF163                            ( * * )
                *
                * LISTING VARIABLE TRUE
                *
EFB1B 0000          CON(5) =aNEG1        -1
          0
EFB20 0000          CON(5) =aTOLST       TOLIST
          0
EFB25 0000          CON(5) =STORE        !            ( * * )
          0
                *
                * PAGE NUMBER TO ONE
                *
EFB2A 0000          CON(5) =ONE          1
          0
EFB2F 0000          CON(5) =aPAGE#       PAGENO
          0
EFB34 0000          CON(5) =STORE        !            ( * * )
          0
                *
                * LISTING LINE NUMBER TO ONE
                *
EFB39 0000          CON(5) =ONE          1
          0
EFB3E 0000          CON(5) =aLLIN#       LLINE#
          0
EFB43 0000          CON(5) =STORE        !            ( * * )
          0
                *
                * NO TITLE YET
                *
EFB48 0000          CON(5) =NULL$        NULL$
          0
EFB4D 0000          CON(5) =aTIT         TIT
          0
EFB52 0000          CON(5) =S!           S!           ( * * )
          0
                *
                * NO SUBTITLE YET
                *
EFB57 0000          CON(5) =NULL$        NULL$
          0
```



```
EFB5C 0000          CON(5) =aSTIT        STIT
          0
EFB61 0000          CON(5) =S!           S!           ( * * )
          0
                *
                * SET LC TO BEGINNING OF OBJECT FILE
                *
EFB66 0000          CON(5) =aOBJFL       OBJFILE
          0
EFB6B 0000          CON(5) =AT           @
          0
EFB70 0000          CON(5) =aLC          LC
          0
EFB75 0000          CON(5) =STORE        !            ( * * )
          0
                *
                * SOURCE LINE TO ONE
                *
EFB7A              %TH163                            THEN
EFB7A 0000          CON(5) =ONE          1
          0
EFB7F 0000          CON(5) =aSLIN#       SLINE#
          0
EFB84 0000          CON(5) =STORE        !            ( * * )
          0
                *
                * DOTS ON DISPLAY TO ZERO
                *
EFB89 0000          CON(5) =ZERO         0
          0
EFB8E 0000          CON(5) =aLINEC       LINECNT
          0
EFB93 0000          CON(5) =STORE        !            ( * * )
          0
                *
                * NO INSTRUCTION REQUIRING GOYES/RTNYES
                *
EFB98 0000          CON(5) =ZERO         0
          0
EFB9D 0000          CON(5) =aLSTRQ       LAST.REQ
          0
EFBA2 0000          CON(5) =STORE        !
          0
                *
                * NOT DONE YET
                *
EFBA7 0000          CON(5) =ZERO         0
          0
EFBAC 0000          CON(5) =aDONE        DONE
          0
EFBB1 0000          CON(5) =STORE        !            ( * * )
          0
                *
                * NO ENTRY PSEUDO-OPS
                *
```

```
590 EFBB6 0000          CON(5) =ZERO            0
          0
591 EFBBB 0000          CON(5) =aTENT           TENT
          0
592 EFBC0 0000          CON(5) =STORE           !           ( * * )
          0
593             *
594             * NO KEYH PSEUDO-OPS
595             *
596 EFBC5 0000          CON(5) =ZERO            0
          0
597 EFBCA 0000          CON(5) =aKENT           KENT
          0
598 EFBCF 0000          CON(5) =STORE           !
          0
599 EFBD4 0000          CON(5) =SEMI            ;           ( * * )
          0
600             ****************************************************************
601             ***
602             ***  PURGE.ALL
603             ***
604             *** ( -- )
605             *** ( ENSURE ALL REQ'D FILES PURGED )
606             ****************************************************************
607 EFBD9 0000 =aPUALL CON(5) =DOCOL           : PURGE.ALL
          0
608             *
609             * PURGE OLD INTERNAL OBJECT FILES
610             *
611 EFBDE 0000          CON(5) =QUOTC           " ASMBOB"
          0
612 EFBE3 60            CON(2) 6
613 EFBE5 60            CON(2) 6
614 EFBE7 1435          NIBASC \ASMBOB\
          D424
          F424
615 EFBF3 0000          CON(5) =aREMOV          REMOVE
          0
616 EFBF8 0000          CON(5) =DROP            DROP        ( * * )
          0
617             *
618             * PURGE OLD INTERNAL SYMBOL TABLES
619             *
620 EFBFD 0000          CON(5) =QUOTC           " ASMBSY"
          0
621 EFC02 60            CON(2) 6
622 EFC04 60            CON(2) 6
623 EFC06 1435          NIBASC \ASMBSY\
          D424
          3595
624 EFC12 0000          CON(5) =aREMOV          REMOVE
          0
625 EFC17 0000          CON(5) =DROP            DROP        ( * * )
          0
626             *
```



```
627             * REMOVE OLD LISTING FILES, IF THEY'RE PRESENT
628             *
629 EFC1C 0000          CON(5) =LISTNG          LISTING
          0
630 EFC21 0000          CON(5) =FSTX            FSYNTAX
          0
631 EFC26 0000          CON(5) =ZBRNH           IF
          0
632 EFC2B 0300          REL(5) %IF164
          0
633 EFC30 0000          CON(5) =LISTNG          LISTING
          0
634 EFC35 0000          CON(5) =aREMOV          REMOVE      ( * F * )
          0
635 EFC3A 0000          CON(5) =ZEQ             0=
          0
636 EFC3F 0000          CON(5) =ZBRNH           IF
          0
637 EFC44 3200          REL(5) %IF165
          0
638             *
639             * COULDN'T PURGE LISTING FILE BECAUSE IT WASN'T TEXT
640             *
641 EFC49 0000          CON(5) =LIT             48
          0
642 EFC4E 0300          CON(5) 48               'listing file not TEXT'
          0
643 EFC53 0000          CON(5) =aERMSG          ERROR MESSAGE
          0
644 EFC58 0000          CON(5) =CR              CR
          0
645 EFC5D 0000          CON(5) =TYPE            TYPE
          0
646 EFC62 0000          CON(5) =ABORT           ABORT
          0
647 EFC67        %IF165                         THEN
648 EFC67        %IF164                         THEN
649 EFC67 0000          CON(5) =SEMI            ;           ( * * )
          0
650             ****************************************************************
651             ***
652             ***  INIT
653             ***
654             *** ( STR -- )
655             *** ( INITIALIZE THE ASSEMBLER TO RUN )
656             ****************************************************************
657 EFC6C 0000 =aINIT  CON(5) =DOCOL           : INIT
          0
658             *
659             * ENSURE SOURCE <> LISTING
660             *
661 EFC71 0000          CON(5) =2DUP            2DUP
          0
662 EFC76 0000          CON(5) =LISTNG          LISTING
          0
```

```
663 EFC7B 0000          CON(5) =S=              S=          ( * STR F * )
          0
664 EFC80 0000          CON(5) =ZBRNH           IF
          0
665 EFC85 3200          REL(5) %IF151
          0
666 EFC8A 0000          CON(5) =LIT             49
          0
667 EFC8F 1300          CON(5) 49               'invalid listing file'
          0
668 EFC94 0000          CON(5) =aERMSG          ERROR MESSAGE
          0
669 EFC99 0000          CON(5) =CR              CR
          0
670 EFC9E 0000          CON(5) =TYPE            TYPE
          0
671 EFCA3 0000          CON(5) =ABORT           ABORT
          0
672 EFCA8       %IF151                          THEN        ( * STR * )
673             *
674             * SAVE SOURCE FILE NAME
675             *
676 EFCA8 0000          CON(5) =LIT             20
          0
677 EFCAD 4100          CON(5) 20
          0
678 EFCB2 0000          CON(5) =MIN             MIN
          0
679 EFCB7 0000          CON(5) =a>PAD2          >PAD2       ( * * )
          0
680             *
681             * PURGE EXISTING FILES
682             *
683 EFCBC 9DBF          CON(5) =aPUALL          PURGE.ALL   ( * * )
          E
684             *
685             * TURN OFF ATTN KEY
686             *
687 EFCC1 0000          CON(5) =LIT             15
          0
688 EFCC6 F000          CON(5) 15
          0
689 EFCCB 0000          CON(5) =LIT             193601
          0
690 EFCD0 144F          CON(5) 193601
          2
691 EFCD5 0000          CON(5) =N!              N!          ( * * )
          0
692             *
693             * INITIALIZE VARIABLES
694             *
695 EFCDA 0000          CON(5) =aVARIN          VAR.INIT    ( * * )
          0
696             *
697             * SAVE SOURCE FILE
```



```
698             *
699 EFCDF 0000          CON(5) =aPAD2>          PAD2>       ( * STR * )
          0
700 EFCE4 0000          CON(5) =aSFILE          SRC.FILE
          0
701 EFCE9 0000          CON(5) =S!              S!          ( * * )
          0
702             *
703             * INITIALIZE OBJECT FILE
704             *
705 EFCEE 0000          CON(5) =aOBJIN          OBJ.INIT
          0
706             *
707             * INITIALIZE SYMBOL TABLE
708             *
709 EFCF3 0000          CON(5) =aSYTIN          SYT.INIT
          0
710             *
711             * INITIALIZE LISTING FILE
712             *
713 EFCF8 0000          CON(5) =aLSTIN          LIST.INIT
          0
714             *
715             * OPEN SOURCE FILE
716             *
717 EFCFD 0000          CON(5) =aSFILE          SFILE       ( * STR * )
          0
718 EFD02 0000          CON(5) =aOPSRC          OPEN.SOURCE
          0
719 EFD07 0000          CON(5) =SEMI            ;           ( * * )
          0
720             ****************************************************************
721             ***
722             ***  DO.PASS
723             ***
724             *** ( -- )
725             *** ( DO ONE PASS )
726             ****************************************************************
727 EFD0C 0000 =aDOPAS CON(5) =DOCOL           : DO.PASS
          0
728 EFD11 5DAF          CON(5) =aPASIN          PASS.INIT   ( * * )
          E
729             *
730             * while <NOT EOF> and <NOT DONE> do
731             *
732 EFD16        %BG18                          BEGIN       ( * * )
733 EFD16 0000          CON(5) =aRDSRC          READ.SOURCE
          0
734 EFD1B 0000          CON(5) =aDONE           DONE
          0
735 EFD20 0000          CON(5) =AT              @
          0
736 EFD25 0000          CON(5) =ZEQ             0=
          0
737 EFD2A 0000          CON(5) =AND             AND         ( * F * )
```

```
          0
738 EFD2F 0000          CON(5) =ZBRNH           WHILE
          0
739 EFD34 4100          REL(5) %RP18
          0
740               *
741               * PROCESS A LINE
742               *
743 EFD39 71AF          CON(5) =aDOLIN          DO.LINE      ( * * )
          E
744 EFD3E 0000          CON(5) =BRNCH           REPEAT
          0
745 EFD43 3DFF          REL(5) %BG18
          F
746               *
747               * end while
748               *
749 EFD48        %RP18
750 EFD48 0000          CON(5) =SEMI            ;            ( * * )
          0
751               ****************************************************************
752               ***
753               ***  no more 8 char words
754               ***
755               ****************************************************************
756 EFD4D 0000          CON(5) 0                Link: no more 8 char words
          0
757 EFD52 88    =!ASSEM CON(2) #80+8
758 EFD54 1435          NIBASC \ASSEMBL\
          3554
          D424
          C4
759 EFD62 5C            CON(2) \E\+#80
760 EFD64 0000 =aASSEM  CON(5) =DOCOL           : ASSEMBLE
          0
761               *** ( STR -- )
762               *** ( WORD WHICH EVOKES THE SASM ASSEMBLER )
763               *
764               * ENSURE WE'RE NOT EXECUTING FORTHX
765               *
766 EFD69 0000          CON(5) =BASIC           BASIC
          0
767 EFD6E 0000          CON(5) =AT              @
          0
768 EFD73 0000          CON(5) =ZBRNH           IF           ( * STR * )
          0
769 EFD78 8200          REL(5) %IF5
          0
770 EFD7D 0000          CON(5) =LIT             63
          0
771 EFD82 F300          CON(5) 63               'BASIC not re-entrant'
          0
772 EFD87 0000          CON(5) =aERMSG          ERROR MESSAGE
          0
773 EFD8C 0000          CON(5) =CR              CR
```



```
          0
774 EFD91 0000          CON(5) =TYPE            TYPE
          0
775 EFD96 0000          CON(5) =CR              CR
          0
776 EFD9B 0000          CON(5) =ABORT           ABORT
          0
777 EFDA0        %IF5                           THEN         ( * STR * )
778               *
779               * SAVE CURRENT VALUE OF BASE
780               *
781 EFDA0 0000          CON(5) =BASE            BASE
          0
782 EFDA5 0000          CON(5) =AT              @
          0
783 EFDAA 0000          CON(5) =CSP             CSP
          0
784 EFDAF 0000          CON(5) =STORE           !
          0
785               *
786               * INITIALIZE VARIABLES & FILES
787               *
788 EFDB4 C6CF          CON(5) =aINIT           INIT         ( * * )
          E
789               *
790               * DO PASS 1
791               *
792 EFDB9 0000          CON(5) =ONE             1
          0
793 EFDBE 0000          CON(5) =aPASS           PASS
          0
794 EFDC3 0000          CON(5) =STORE           !
          0
795 EFDC8 C0DF          CON(5) =aDOPAS          DO.PASS      ( * * )
          E
796               *
797               * CHECK IF USER SUPPLIED THE LABEL "FiLeNd"
798               *
799 EFDCD 0000          CON(5) =QUOTC           " FiLeNd"
          0
800 EFDD2 60            CON(2) 6
801 EFDD4 60            CON(2) 6
802 EFDD6 6496          NIBASC \FiLeNd\
          C456
          E446
803 EFDE2 0000          CON(5) =aSYT>           SYT>         ( * NUM * )
          0
804 EFDE7 0000          CON(5) =ZBRNH           IF
          0
805 EFDEC 4100          REL(5) %IF159
          0
806 EFDF1 0000          CON(5) =LIT             61
          0
807 EFDF6 D300          CON(5) 61               'restricted label FiLeNd exists'
          0
```

```
808 EFDFB 0000          CON(5) =aSHUTD          SHUTDOWN
          0
809 EFE00       %IF159                          THEN        ( * * )
810             *
811             * PUT LABEL FiLeNd ONTO THE SYMBOL TABLE
812             *
813 EFE00 0000          CON(5) =aLC             LC
          0
814 EFE05 0000          CON(5) =AT              @
          0
815 EFE0A 0000          CON(5) =DUP             DUP
          0
816 EFE0F 0000          CON(5) =QUOTC           "
          0
817 EFE14 60            CON(2) 6
818 EFE16 60            CON(2) 6
819 EFE18 6496          NIBASC \FiLeNd\
          C456
          E446
820 EFE24 0000          CON(5) =a>SYT           >SYT        ( * LC * )
          0
821             *
822             * CALCULATE & STORE FILE SIZE IN NIBBLES
823             *
824 EFE29 0000          CON(5) =aOBJFL          OBJFILE
          0
825 EFE2E 0000          CON(5) =AT              @
          0
826 EFE33 0000          CON(5) =MINUS           -
          0
827 EFE38 0000          CON(5) =aFLSIZ          FILESIZE
          0
828 EFE3D 0000          CON(5) =STORE           !           ( * * )
          0
829             *
830             * RESET THE SOURCE FILE TO BEGINNING
831             *
832 EFE42 0000          CON(5) =aRSSRC          RESET.SRC   ( * * )
          0
833             *
834             * DO PASS 2
835             *
836 EFE47 0000          CON(5) =TWO             2
          0
837 EFE4C 0000          CON(5) =aPASS           PASS
          0
838 EFE51 0000          CON(5) =STORE           !
          0
839 EFE56 C0DF          CON(5) =aDOPAS          DO.PASS     ( * * )
          E
840             *
841             * OUTPUT THE SYMBOL TABLE
842             *
843 EFE5B 0000          CON(5) =aTOLST          TOLIST
          0
```



```
844 EFE60 0000          CON(5) =AT              @           ( * F * )
          0
845 EFE65 0000          CON(5) =ZBRNH           IF
          0
846 EFE6A A000          REL(5) %IF166
          0
847 EFE6F 0000          CON(5) =aOUTSY          OUT.SYT
          0
848 EFE74       %IF166                          THEN        ( * * )
849             *
850             * PURGE SYMBOL TABLE & CLOSE FILES
851             *
852 EFE74 0000          CON(5) =aCLEAN          CLEANUP
          0
853 EFE79 0000          CON(5) =SEMI            ;           ( * * )
          0
```

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER

```
                        STITLE FIT STATS
           zSIZE        EQU    (*)-zTHIS
           zLEFT        EQU    (zNEXT)-*
EFE7E                   BSS    zLEFT
```



```
                        STITLE HOOK ROUTINE JUMP TABLES
EFF00                   BSS    #EFF00-*
EFF00 0000              CON(5) =a?PAS1        ?PASS=1
      0
EFF05 0000              CON(5) =aWRUOL        WRAP.UP.OLD
      0
EFF0A 0000              CON(5) =aSIMLN        SIM.LINE
      0
EFF0F 0000              CON(5) =aSTUNW        START.UP.NEW
      0
EFF14 0000              CON(5) =a>SYT         >SYT
      0
EFF19 0000              CON(5) =aSYT>         SYT>
      0
EFF1E 0000              CON(5) =aERROR        ERROR
      0
EFF23 0000              CON(5) =a>PAD         >PAD
      0
EFF28 0000              CON(5) =aEXPRS        EXPRESSION
      0
EFF2D                   BSS    #EFF80-*
EFF80 0000              CON(5) =aLAB.F        LABEL.FIELD
      0
EFF85 0000              CON(5) =aOPC.F        OPCODE.FIELD
      0
EFF8A 0000              CON(5) =aEXP.F        EXPRESSION.FIELD
      0
EFF8F 0000              CON(5) =aFLTYP        FILETYPE
      0
EFF94 0000              CON(5) =aKNOWN        KNOWN
      0
EFF99 0000              CON(5) =aOLDEX        OLD.EXPRESSION
      0
EFF9E                   END
```

=!ASSEM	Abs	982354	#EFD52	-	757						
%BG17	Abs	980485	#EF605	-	27	98					
%BG18	Abs	982294	#EFD16	-	732	745					
%BG19	Abs	980568	#EF658	-	51	61					
%IF150	Abs	980688	#EF6D0	-	96	39					
%IF151	Abs	982184	#EFCA8	-	672	665					
%IF152	Abs	980673	#EF6C1	-	89	79					
%IF153	Abs	980802	#EF742	-	143	122					
%IF154	Abs	981202	#EF8D2	-	282	273					
%IF155	Abs	981282	#EF922	-	316	298					
%IF156	Abs	981282	#EF922	-	315	308					
%IF157	Abs	981632	#EFA80	-	470	465					
%IF158	Abs	981357	#EF96D	-	351	334					
%IF159	Abs	982528	#EFE00	-	809	805					
%IF160	Abs	981522	#EFA12	-	420	368	398				
%IF161	Abs	981432	#EF9B8	-	386	377					
%IF162	Abs	981497	#EF9F9	-	411	390					
%IF163	Abs	981787	#EFB1B	-	524	507					
%IF164	Abs	982119	#EFC67	-	648	632					
%IF165	Abs	982119	#EFC67	-	647	637					
%IF166	Abs	982644	#EFE74	-	848	846					
%IF167	Abs	981592	#EFA58	-	452	450					
%IF304	Abs	981132	#EF88C	-	254	239					
%IF311	Abs	981672	#EFAA8	-	482	458					
%IF5	Abs	982432	#EFDA0	-	777	769					
%RP17	Abs	980698	#EF6DA	-	102	30					
%RP18	Abs	982344	#EFD48	-	749	739					
%RP19	Abs	980603	#EF67B	-	65	54					
%TH163	Abs	981882	#EFB7A	-	565	523					
%TH304	Abs	981152	#EF8A0	-	262	250					
%TH311	Abs	981682	#EFAB2	-	488	475					
2DROP	Ext			-	412						
2DUP	Ext			-	387	661					
?TERM	Ext			-	28	52					
ABORT	Ext			-	646	671	776				
ADD	Ext			-	258						
AND	Ext			-	76	737					
AT	Ext			-	116	136	233	263	269	296	305	332
					345	373	444	472	559	735	767	782
					814	825	844					
BASE	Ext			-	781						
BASIC	Ext			-	766						
BRNCH	Ext			-	60	97	249	522	744			
CR	Ext			-	43	126	644	669	773	775		
CSP	Ext			-	783							
DOCOL	Ext			-	23	111	156	226	291	330	360	429
					504	607	657	727	760			
DROP	Ext			-	59	246	366	413	616	625		
DUP	Ext			-	70	74	234	815				
Dx=HX	Ext			-	190							
EMIT	Ext			-	71	146						
EQUAL	Ext			-	37	77	271	395	447			
EXEC	Ext			-	264							
F.AND	Ext			-	306							
FSTX	Ext			-	630							

GT	Ext			-	237						
KEY	Ext			-	34	58	69				
LISTNG	Ext			-	629	633	662				
LIT	Ext			-	35	72	83	117	144	230	235	247
					255	309	404	414	445	476	490	511
					518	641	666	676	687	689	770	806
MIN	Ext			-	678						
MINUS	Ext			-	826						
MOD	Ext			-	119						
MULT	Ext			-	257						
N!	Ext			-	691						
NULL$	Ext			-	546	552					
ONE	Ext			-	147	270	534	540	566	792		
PDOTQ	Ext			-	44						
PSTOR	Ext			-	149	347						
QUOTC	Ext			-	130	611	620	799	816			
ROT	Ext			-	75							
S!	Ext			-	548	554	701					
S=	Ext			-	663							
SEMI	Ext			-	103	150	157	283	322	352	382	407
					421	495	599	649	719	750	853	
SPACE	Ext			-	139							
STORE	Ext			-	95	278	340	494	514	521	530	536
					542	561	568	574	580	586	592	598
					784	794	828	838				
SWAP	Ext			-	245	365						
TWO	Ext			-	836							
TYPE	Ext			-	134	138	645	670	774			
ZBRNH	Ext			-	29	38	53	78	121	238	272	297
					307	333	367	376	389	397	449	457
					464	474	506	631	636	664	738	768
					804	845						
ZEQ	Ext			-	120	300	396	448	463	473	635	736
ZERO	Ext			-	93	572	578	584	590	596		
a>PAD	Ext			-	867							
a>PAD2	Ext			-	679							
a>SYT	Ext			-	381	820	864					
a?PAS1	Ext			-	375	456	505	860				
=aASSEM	Abs	982372	#EFD64	-	760							
aBIN	Ext			-	214							
=aBLANK	Abs	980837	#EF765	-	156	171	177	178				
aBRANC	Ext			-	175							
aBSS	Ext			-	193							
aCHAIN	Ext			-	216							
aCHAR	Ext			-	210							
aCLEAN	Ext			-	852							
aDATAT	Ext			-	186							
=aDOLIN	Abs	981527	#EFA17	-	429	743						
aDONE	Ext			-	585	734						
=aDOPAS	Abs	982284	#EFD0C	-	727	795	839					
aDPARI	Ext			-	185							
aEJECT	Ext			-	195							
aEND	Ext			-	196							
aENDTX	Ext			-	217							
aENTRY	Ext			-	209							

```
 aEQU    Ext                 -  197
 aERMSG  Ext                 -  643   668   772
 aERROR  Ext                 -  314   406   416   866
 aEXP.F  Ext                 -  872
 aEXPRS  Ext                 -  868
 aFLSIZ  Ext                 -  827
 aFLTYP  Ext                 -  471   873
 aFORTH  Ext                 -  215
 aHITOK  Ext                 -  520
 aIDN    Ext                 -  205
=aINIT   Abs  982124 #EFC6C  -  657   788
 aISGYS  Ext                 -  299
 aKENT   Ext                 -  597
 aKEYP   Ext                 -  211
 aKNOWN  Ext                 -  874
 aLAB.F  Ext                 -  364   374   870
=aLABLS  Abs  981362 #EF972  -  360   451
 aLABOK  Ext                 -  388
 aLC     Ext                 -  346   372   560   813
=aLC+    Abs  981297 #EF931  -  330   489
 aLCASC  Ext                 -  192
 aLCHEX  Ext                 -  189
 aLENST  Ext                 -  462
 aLEX    Ext                 -  204
 aLINEC  Ext                 -   94   115   148   573
 aLIST   Ext                 -  198
 aLLIN#  Ext                 -  541
 aLOTOK  Ext                 -  513
 aLSTIN  Ext                 -  713
 aLSTRQ  Ext                 -  295   339   579
=aMONAT  Abs  980480 #EF600  -   23   433
 aMSG    Ext                 -  207
 aNEG1   Ext                 -  528
 aNIBFF  Ext                 -  191
 aNIBHX  Ext                 -  187
 aNUM#1  Ext                 -  137
 aOBJFL  Ext                 -  558   824
 aOBJIN  Ext                 -  705
 aOCCUR  Ext                 -  493
 aOLDEX  Ext                 -  875
 aOPC.F  Ext                 -  871
 aOPLEN  Ext                 -  277   304   331   344
 aOPSRC  Ext                 -  718
 aOPTYP  Ext                 -  232   443
 aOUTLS  Ext                 -  321
 aOUTOB  Ext                 -  320
 aOUTSY  Ext                 -  847
 aPAD2>  Ext                 -  699
 aPAGE#  Ext                 -  535
 aPARSE  Ext                 -  438
=aPASIN  Abs  981717 #EFAD5  -  504   728
 aPASS   Ext                 -  135   268   793   837
 aPOLL   Ext                 -  208
=aPROCS  Abs  981047 #EF837  -  226   466   483
 aPTRTE  Ext                 -  180
```



```
=aPUALL  Abs  981977 #EFBD9  -  607   683
 aRDSRC  Ext                 -  733
 aREGAR  Ext                 -  173
 aREGLO  Ext                 -  174
 aREGTE  Ext                 -  172
 aREMOV  Ext                 -  615   624   634
 aRQYES  Ext                 -  338
 aRSSRC  Ext                 -  832
 aRTNYS  Ext                 -  179
 aSETPT  Ext                 -  183
 aSETST  Ext                 -  184
 aSFILE  Ext                 -  700   717
=aSHOWP  Abs  980703 #EF6DF  -  111   434
 aSHUTD  Ext                 -   85   478   808
 aSIMLN  Ext                 -  862
 aSLIN#  Ext                 -  567
=aSPIT   Abs  981207 #EF8D7  -  291   484
 aSTATT  Ext                 -  181
 aSTIT   Ext                 -  553
 aSTITL  Ext                 -  201
 aSTUNW  Ext                 -  863
 aSYT>   Ext                 -  394   803   865
 aSYTIN  Ext                 -  709
=aTABLE  Abs  980847 #EF76F  -  170   231
 aTENT   Ext                 -  591
 aTESTF  Ext                 -  492
 aTIT    Ext                 -  547
 aTITLE  Ext                 -  199
 aTOKNP  Ext                 -  213
 aTOLST  Ext                 -  529   843
 aVARIN  Ext                 -  695
 aWORDI  Ext                 -  203
 aWORDN  Ext                 -  202
 aWRUOL  Ext                 -  861
 zLEFT   Abs     130 #00082  -  856   857
 zNEXT   Abs  982784 #EFF00  -    5   856
 zSIZE   Abs    2174 #0087E  -  855
 zTHIS   Abs  980480 #EF600  -    4   855
```

Input Parameters

  Source file name is GN&AS8

  Listing file name is GN/AS8::65

  Object file name is GN%AS8::65

```
                                111111
                      0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

OFFICIALLY UNOFFICIAL

NOMAS

THE READER AGREES NOT TO
CONTACT THE MANUFACTURER

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.