FORTH AGE

HP-75C FORTH

Users Manual

JOHN CASSADY

#20281

HEWLETT-PACKARD

HP-75C FORTH

Manual

John Cassady

Implementation
of Kernel
Dave Conklin

Printed in U.S.A.

CONTENTS

Introduction 3
Loading Existing Screens 4
Miscellaneous 5
Editor 5
Debugger 6
Size Change 7
HP-75C Machine 7
FORTH Assembler 10
FORTH Disassembler 11
Source Listing of Kernel 13
Kernel Glossary 33
Extensions Screens 43
Extensions Glossary 51

ILLUSTRATIONS

List of Files 3
Relative Location of FORTH 4
DEBUG Example 7
CPU Registers and Flags 8
75C Machine Code 9
75C Memory Layout 10
Disassembler Example 11

NOTICE

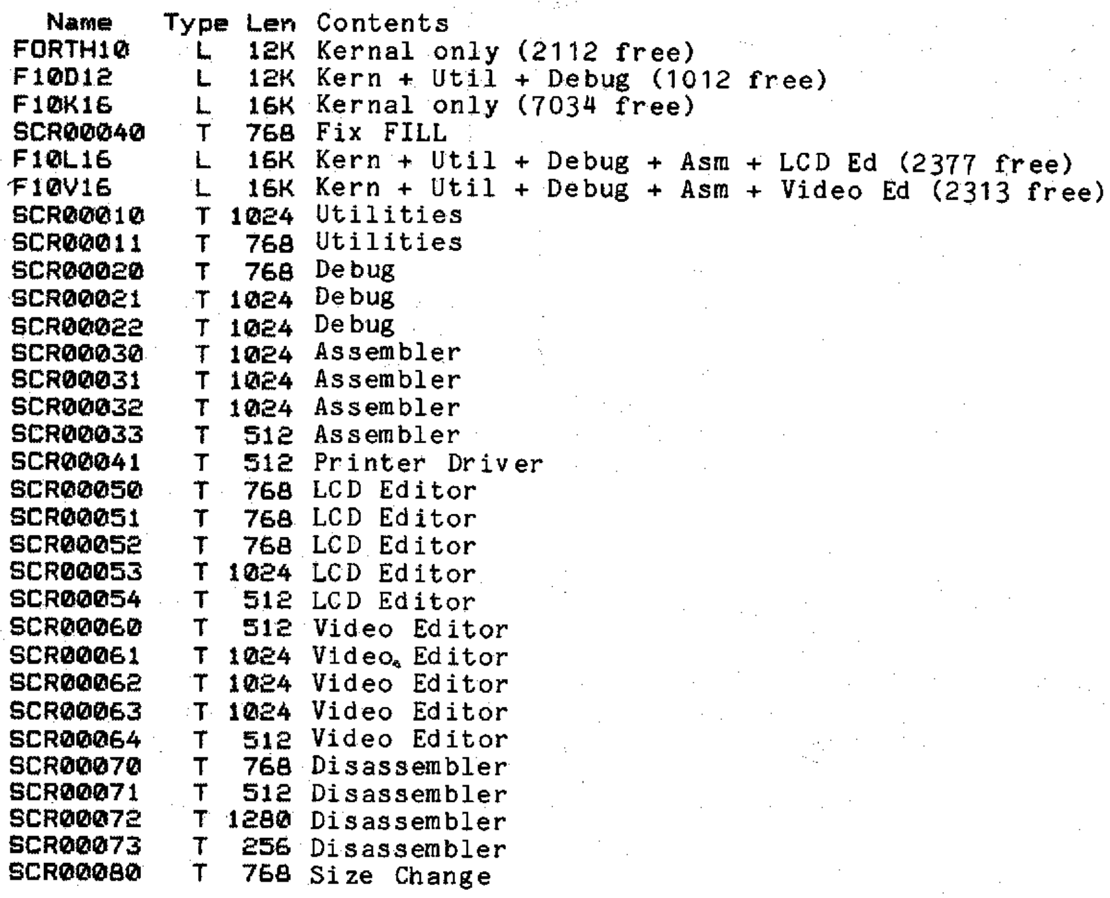
Authors and publisher make no express or implied warranty with regard to the program material offered or the merchantability or the fitness of the program material for any particular purpose. The program material is made available solely on an "as is" basis, and the entire risk as to its quality and performance is with the user. Should the program material prove defective, the user shall bear the entire cost of all necessary correction and all incidental or consequential damages in connection with or arising out the furnishing, use, or performance of the program material.

The material in this manual has not been produced by Hewlett-Paackard Company or any of its affiliates, and they are not responsible for any errors or omissions. The publisher of this material is not connected with Hewlett-Packard Company.

Published April 1983
Forth Age Productions
23 Altarinda Road, Suite 213
Orinda, California 94563
(415) 254-5221

INTRODUCTION

The digital cassette supplied with this manual contains the following files.



```
  Name    Type Len  Contents
FORTH10     L   12K Kernal only (2112 free)
F10D12      L   12K Kern + Util + Debug (1012 free)
F10K16      L   16K Kernal only (7034 free)
SCR00040    T   768 Fix FILL
F10L16      L   16K Kern + Util + Debug + Asm + LCD Ed (2377 free)
F10V16      L   16K Kern + Util + Debug + Asm + Video Ed (2313 free)
SCR00010    T  1024 Utilities
SCR00011    T   768 Utilities
SCR00020    T   768 Debug
SCR00021    T  1024 Debug
SCR00022    T  1024 Debug
SCR00030    T  1024 Assembler
SCR00031    T  1024 Assembler
SCR00032    T  1024 Assembler
SCR00033    T   512 Assembler
SCR00041    T   512 Printer Driver
SCR00050    T   768 LCD Editor
SCR00051    T   768 LCD Editor
SCR00052    T   768 LCD Editor
SCR00053    T  1024 LCD Editor
SCR00054    T   512 LCD Editor
SCR00060    T   512 Video Editor
SCR00061    T  1024 Video Editor
SCR00062    T  1024 Video Editor
SCR00063    T  1024 Video Editor
SCR00064    T   512 Video Editor
SCR00070    T   768 Disassembler
SCR00071    T   512 Disassembler
SCR00072    T  1280 Disassembler
SCR00073    T   256 Disassembler
SCR00080    T   768 Size Change
```

Figure 1
List of files, their size, type,
contents and free dictionary space.

These files contain five copies of FORTH, each with different precomplied extensions. FORTH10, and F10D12 will run in a standard HP-75C without the plug-in RAM. Versions F10K16, F10L16 and F10V16 require the additional 8K of RAM provided by the plug-in module. The contents of each of these versions of FORTH is listed in figure 1 together with the amount of dictionary space remaining.

A typical loading sequence will proceed like this. The digital cassette drive is attached to the computer with the HP-IL cables. It has been assigned a name using the ASSIGNIO procedure. We will assume the name given to the cassette drive is ':CA'. If you have the video or printer, they should be the DISPLAY IS and PRINTER IS devices respectively. To load FORTH you will type

COPY 'F10D12:CA' TO 'FORTH'
>   (now type) FORTH
HP75 FORTH 1.0
OK

Your FORTH is now ready to use. To see the list of available words type VLIST. You may return to the 75C operating system by typing BYE. FORTH does not convert lower case to upper case. Therefore, if you define a word using lower case, you must type it in lower case to execute it; The same with upper case.

FORTH is contained in a language extension file (LEX file). Thus, it is a self-contained unit including dictionary, stack and block buffers.

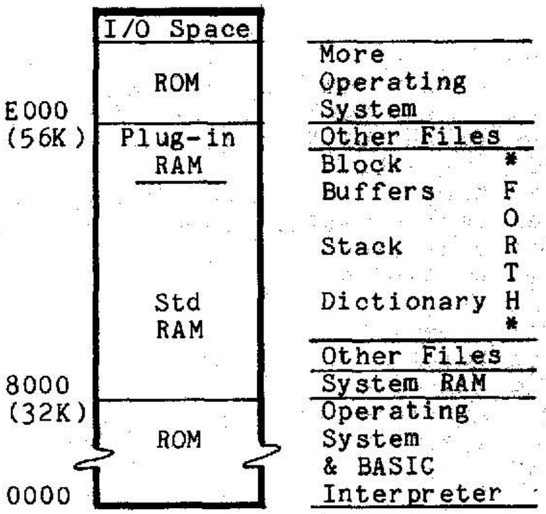


Figure 2
Relative location of FORTH in 75C memory.

Although F10A is copied into RAM as a 12K file, the 12K includes the entire memory space taken up by FORTH: dictionary, unused dictionary space, stack, user area and buffers. For comparison purposes, the "size" of FORTH10 is 6825 bytes — consistent with other Fig FORTH implementations.

All RAM programs in the HP-75C must be relocatable. That is, they must be able to run irrespective of their actual memory location. This requirement is imposed by the operating system which moves files freely whenever new files are created or old files deleted. Hence, FORTH address references are relative. They assume that FORTH is located at memory position zero. When you type ' LIT . you will see the address 133 returned. The PFA of LIT is located 133 bytes from the start of FORTH. If you are interested in the absolute address, type ' LIT KN + U. and you will be seeing an address above 34500. This is the actual location in 75C RAM and will vary depending on other files being resident.

LOADING EXISTING SCREENS

Most of the screens on the digital cassette are already compiled in one or more versions of FORTH. In the event you want to load words that are not precompiled or you want to change the order of the words you may load from the existing screens. There are 26 screens; they are listed in figure 1. We will use the disassembler as an example. In order to load the disassembler you must first copy each disassembler screen into a 75C text file.

```
COPY 'SCR00070:CA' TO 'SCR00070'
COPY 'SCR00071:CA' TO 'SCR00071'
COPY 'SCR00072:CA' TO 'SCR00072'
COPY 'SCR00073:CA' TO 'SCR00073'
```

Now you go into FORTH and commence loading.

```
70 LOAD
71 LOAD
```

Note that screens 72 and 73 of the disassembler are not to be loaded. They are data; they must be resident in memory for the disassembler to run, but they are not compiled.

MISCELLANEOUS

The hardware used in programming the FORTH extensions, not the kernal, consists of the 82161A Digital Cassette Drive, the 82163 Video Interface and a printer consisting of the 82165 HP-IL/GPIO interface and an Okidata 83A. The printer could just as well have been an 82905B Impact Printer.

The persistence of the LCD viewing window on the 75C can be controlled by setting the operating system variable DELAY.

The kernal derives from the FORTH Interest Group Fig-FORTH, adapted for HP-85 and then further adapted for HP-75C. A significant departure is in the implementation of vocabularies. FORTH Interest Group source documents are available through the publisher of this manual.

If your system "hangs up", i.e., crashes, you will have to reset by pressing shift control CLR. This causes memory loss; hence, you must reload FORTH. A problem that may occasionally be encountered is being locked in a loop, either an infinite loop (BEGIN . . . AGAIN) or a (DO . . . LOOP) with erroneous indices. This fault may be solved by pressing the ATTN key.

A word of caution regarding FORGET. When you use FORGET, the CURRENT and CONTEXT vocabularies must be the same and the word you are FORGETting must be in that vocabulary. For example, you are loading the assembler (75ASM vocabulary) and half way through you change your mind. You type FORGET 75ASM — crash— 75ASM is in FORTH. FORGET does not unscramble your vocabulary threads. When using multiple vocabularies, use FORGET with caution.

Another potential pitfall is defining a word without a name. For example (do not do these examples—
CRASH!!!)

```
1234 CONSTANT (rtn)
1234 VARIABLE (rtn)
12 USER (rtn)
VOCABULARY (rtn)
: ; (rtn)
```

In every case we have redefined null. Null is, however, a word in the FORTH system, which may be seen on page 35 of the source listing. The new definition hides the old one, but the old one is required for the interpreter to function correctly. Hence, the system is unable to function; the computer must be reset and FORTH reloaded. Other problems usually involve storing into an improper memory location, incorrect CMOVEs and FILLs. With normal care the 75C FORTH is robust and highly crash-resistant.

EDITOR

In order to create your own source screens or to modify existing screens you must use an editor. There are three editors available. Since FORTH source screens are contained in 75C text files, they can be edited using the resident 75C text editor. Instructions are found in the 75C Owners Manual. When using this method you must observe the restriction of 16 lines and 64 characters per line. The actual line numbers are irrelevant.

When creating your own screens you must first open a text file using the 75C operating system. For example, you want to write a program by editing it into screen 101. Return to the 75C operating system and type

```
EDIT 'SCR00101',TEXT
:10.....
```

The purpose of the 10..... is to give the file a line. Without a line with text it is an empty file and may be automatically purged. With this file in existence return to FORTH and type

```
101 EDIT
```

You are now in edit mode. The video screen or display will clear. The dots and the cursor will appear. You can now begin entering your program, typing over the dots. The details of editing are covered below.

Assume that your source has been typed in and edited to your satisfaction. You want to save the screen on cassette or magnetic card. To save it type ESC (control BACK). This will copy the FORTH block buffer containing your screen back to the operating system text file having the name which corresponds to your screen number. If you want to leave the editor without copying the screen back to the text file, then type shift control TAB (hold down the shift and control keys, then press TAB).

To copy the screen to cassette or card you must leave FORTH and returning to the operating system by typing BYE, then type

```
COPY 'SCR00101' TO 'SCR00101:CA'
```

to copy it to cassette or

```
COPY 'SCR00101' TO CARD
```

to copy it to magnetic card. Any screen will fit on a single magnetic card.

A full cursor-controlled screen-type editor is provided in two versions. One is customized for use with the built-in LCD. The other is adapted to use the features of the 16 line 32 character HP-IL video. You will be using one editor or the other. The LCD editor is contained in F10L16; the video editor in F10V16. The commands are identical; the only differences being those imposed by the respective display technologies.

The easiest way to become familiar with the FORTH editor is to try it. Copy screen 11 to a text file from cassette.

```
COPY 'SCR00011:CA' TO 'SCR00011'
```

Return to FORTH and type

```
11 EDIT
```

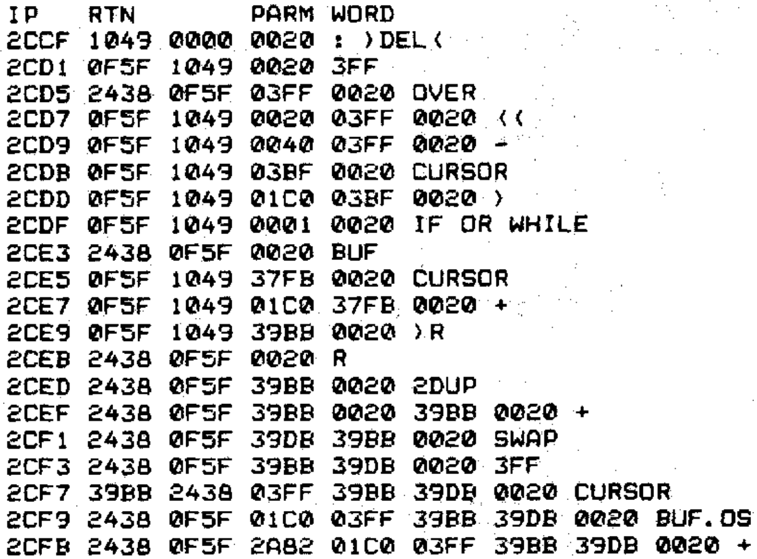
If you are using F10L16 you will see line 0 of screen 11 in the LCD. If you are using F10V16 and the video display you will see the top 16 lines of screen 11 displayed. For available functions look at the listing in this manual of screen 53 for the LCD editor or of screen 63 for the video editor. Try all the functions; their names indicate what they do.

To switch between the upper and lower 16 lines on the video type, control up-arrow or control down-arrow. To leave the editor without saving the screen, type shift control TAB. To leave and save it type ESC (control BACK). Even if you do not save it, it will still be there when you return if you do not leave FORTH or copy in a screen over it by editing another screen which uses the same buffer.

DEBUGGER

DEBUG is a useful tool for creating and analyzing programs in FORTH. DEBUG is adapted from Asprey, T. "A FORTH Execution Simulator for Debugging" Proceeding 1980 FORML Conference, Asilomar, pp 181-187. The debug routine provides all of the functions associated with a decompiler. It also prints on the LCD or video the IP (instruction pointer), two return stack levels and the contents of the parameter stack as well as the name of the words being executed. This routine is called a high level trace because it only lists the high level words in definitions, not the subwords. It may be used with any colon defined word.

By making the printer the DISPLAY IS device, the debug output may be printed. Figure 3 is such a listing from an actual debug session, finding an error in >DEL< .



```
IP   RTN       PARM WORD
2CCF 1049 0000 0020 : )DEL(
2CD1 0F5F 1049 0020 3FF
2CD5 2438 0F5F 03FF 0020 OVER
2CD7 0F5F 1049 0020 03FF 0020 ((
2CD9 0F5F 1049 0040 03FF 0020 -
2CDB 0F5F 1049 03BF 0020 CURSOR
2CDD 0F5F 1049 01C0 03BF 0020 )
2CDF 0F5F 1049 0001 0020 IF OR WHILE
2CE3 2438 0F5F 0020 BUF
2CE5 0F5F 1049 37FB 0020 CURSOR
2CE7 0F5F 1049 01C0 37FB 0020 +
2CE9 0F5F 1049 39BB 0020 )R
2CEB 2438 0F5F 0020 R
2CED 2438 0F5F 39BB 0020 2DUP
2CEF 2438 0F5F 39BB 0020 39BB 0020 +
2CF1 2438 0F5F 39DB 39BB 0020 SWAP
2CF3 2438 0F5F 39BB 39DB 0020 3FF
2CF7 39BB 2438 03FF 39BB 39DB 0020 CURSOR
2CF9 2438 0F5F 01C0 03FF 39BB 39DB 0020 BUF.OS
2CFB 2438 0F5F 2A82 01C0 03FF 39BB 39DB 0020 +
```

Figure 3
Example of the use of DEBUG

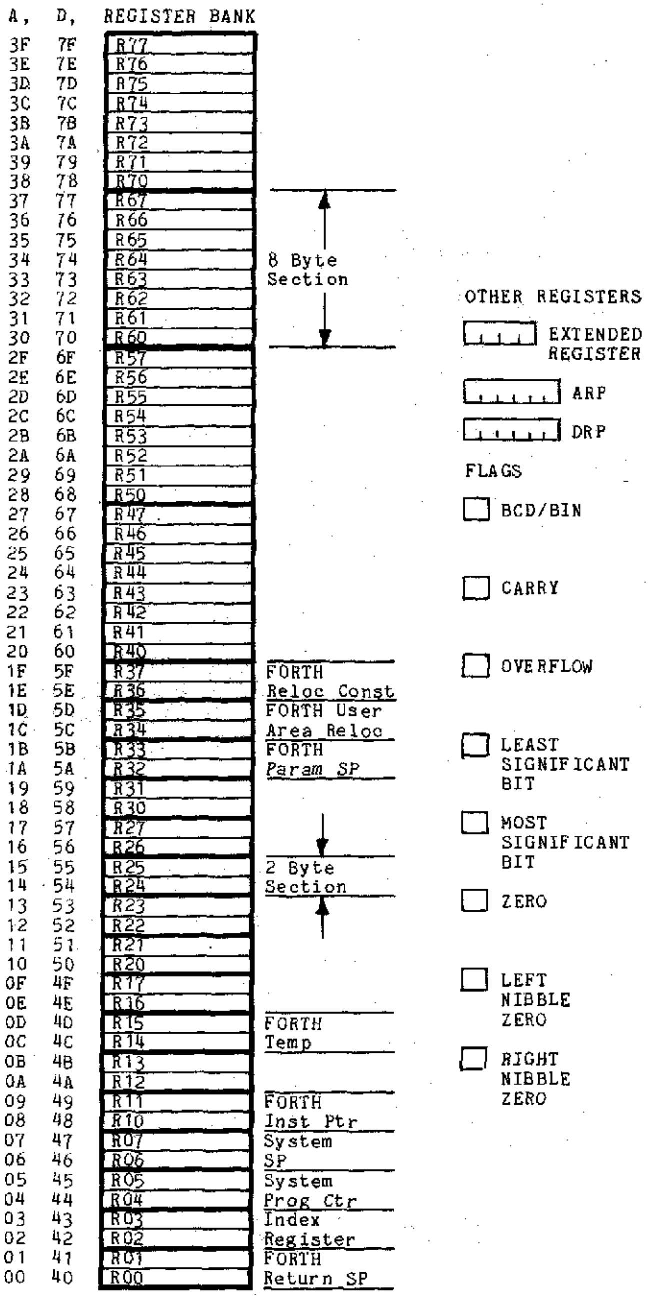
SIZE CHANGE

Screen 80 contains the program used to increase the size of FORTH to 16K. This was done to allow room for the assembler and screen editor. This program can be used to make FORTH larger yet, if desired. With a 16K FORTH there are still 5900 bytes of available RAM. It cannot be used to make FORTH smaller. Do not give SIZE+ a negative or excessively large argument.

HP-75C MACHINE

To make effective use of the FORTH assembler and disassembler, information about the internals of the HP-75C is helpful. The HP-75C uses an eight bit CMOS central processing unit (cpu) which executes an instruction set identical to that of the HP Series 80 computers. The cpu, from the programmers view, consists of a single 64 byte register, two six bit pointers for addressing that register and a set of flags for testing for sign, overflow, etc. The internal makeup is illustrated in figure 4. The 64 byte register is broken up into either groups of two or groups of eight bytes. The lower 32 are grouped in two's, the upper 32 in eight's.

Each register byte has a name. Its name corresponds to its address in octal preceded by an R. Thus, the registers are called R00, R01 . . . R06, R07, R10 . . . R76, R77, the octal digits for the decimal numbers 0 through 64. Even when everything else is done in hex, the registers retain their octal designations. Registers R00 through R37 are broken into groups of two and used for addressing and for eight and sixteen bit arithmetic. The first few of them are, however, dedicated to specific functions. Registers R04/05 are the program counter, R06/07 the machine stack pointer, R02/03 the index register. The registers used by FORTH are listed on page 1 of the source listing. The remaining registers, R40 through R77, are broken into groups of eight and used for floating point arithmetic, string manipulation, etc. Any register or group of registers can be used as an accumulator and any of the two byte groups as a stack pointer.



A,	D,	REGISTER BANK	
3F	7F	R77	
3E	7E	R76	
3D	7D	R75	
3C	7C	R74	
3B	7B	R73	
3A	7A	R72	
39	79	R71	
38	78	R70	
37	77	R67	
36	76	R66	
35	75	R65	
34	74	R64	8 Byte Section
33	73	R63	
32	72	R62	
31	71	R61	
30	70	R60	
2F	6F	R57	
2E	6E	R56	
2D	6D	R55	
2C	6C	R54	
2B	6B	R53	
2A	6A	R52	
29	69	R51	
28	68	R50	
27	67	R47	
26	66	R46	
25	65	R45	
24	64	R44	
23	63	R43	
22	62	R42	
21	61	R41	
20	60	R40	
1F	5F	R37	FORTH Reloc Const
1E	5E	R36	
1D	5D	R35	FORTH User Area Reloc
1C	5C	R34	
1B	5B	R33	FORTH Param SP
1A	5A	R32	
19	59	R31	
18	58	R30	
17	57	R27	
16	56	R26	
15	55	R25	2 Byte Section
14	54	R24	
13	53	R23	
12	52	R22	
11	51	R21	
10	50	R20	
0F	4F	R17	
0E	4E	R16	
0D	4D	R15	FORTH Temp
0C	4C	R14	
0B	4B	R13	
0A	4A	R12	
09	49	R11	FORTH Inst Ptr
08	48	R10	
07	47	R07	System SP
06	46	R06	
05	45	R05	System Prog Ctr
04	44	R04	
03	43	R03	Index Register
02	42	R02	
01	41	R01	FORTH Return SP
00	40	R00	

Figure 4
CPU Registers and Flags

The instruction set for the 75C uses all but two of the 256 possible single byte instructions. The first 128 opcodes are devoted to loading the two six bit registers that address the large 64 byte register. These are called ARP and DRP for address register pointer and data register pointer. The first 64 opcodes load ARP, the second load DRP. In hex, opcodes 0 through 3F load ARP, 40 through 7F load DRP. Thus, opcode 1E causes ARP to point to register R36. Opcode 1 and 41 are exceptions. They respectively cause ARP or DRP to be loaded with the least significant six bits of R00. The side effect of this is that R01 can only be addressed with a multibyte instruction.

The complete 75C machine code instruction set is presented graphically is figures 5a and 5b.

7	6	5	4	3	2	1	0
0		DR:/ AR:	<>000001 load with literal =000001 load with R00				
1	0	0	0	0	log/ ext	right/ left	m/b
1	0	0	0	1	0	decr/ incr	m/b
1	0	0	0	1	1	9s cmp/ 10s cmp	m/b
1	0	0	1	0	0	clear/ test	m/b
1	0	0	1	0	1	xor/ or	m/b
1	0	0	1	1	000 BIN 001 BCD 010 SAD 011 DCE 100 ICE 101 CLE 110 RTN 111 PAD		
1	0	1	000 reg imm 001 reg dir 010 lit imm 011 reg ind 100 lit dir 101 inx dir 110 lit ind 111 inx ind			store/ load	m/b
1	1	0	00 reg imm 01 lit imm 10 lit dir 11 reg dir		00 cmp 01 add 10 sub		m/b
					11 and		1
1	1	0	00 inx 01 lit		11 JSB		0
1	1	1	0	ind/ dir	push/ pop	-adr/ +adr	m/b
1	1	1	1	000 001 010 011 100 101 110 111			JNO/JMP JEV/JOD JPS/JNG JZR/JNZ JEZ/JEN JCY/JNC JLN/JLZ JRN/JRZ

X/Y = 1/0

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	A,00	A,01*	A,02	A,03	A,04	A,05	A,06	A,07	A,10	A,11	A,12	A,13	A,14	A,15	A,16	A,17
1	A,20	A,21	A,22	A,23	A,24	A,25	A,26	A,27	A,30	A,31	A,32	A,33	A,34	A,35	A,36	A,37
2	A,40	A,41	A,42	A,43	A,44	A,45	A,46	A,47	A,50	A,51	A,52	A,53	A,54	A,55	A,56	A,57
3	A,60	A,61	A,62	A,63	A,64	A,65	A,66	A,67	A,70	A,71	A,72	A,73	A,74	A,75	A,76	A,77
4	D,00	D,01*	D,02	D,03	D,04	D,05	D,06	D,07	D,10	D,11	D,12	D,13	D,14	D,15	D,16	D,17
5	D,20	D,21	D,22	D,23	D,24	D,25	D,26	D,27	D,30	D,31	D,32	D,33	D,34	D,35	D,36	D,37
6	D,40	D,41	D,42	D,43	D,44	D,45	D,46	D,47	D,50	D,51	D,52	D,53	D,54	D,55	D,56	D,57
7	D,60	D,61	D,62	D,63	D,64	D,65	D,66	D,67	D,70	D,71	D,72	D,73	D,74	D,75	D,76	D,77
8	ELB	ELM	ERB	ERM	LLB	LLM	LRB	LRM	ICB	ICM	DCB	DCM	TCB	TCM	NCB	NCM
9	TSB	TSM	CLB	CLM	ORB	ORM	XRB	XRM	BIN	BCD	SAD	DCE	ICE	CLE	RTN	PAD
A	LDB	LDM	STB	STM	LDBD	LDMD	STBD	STMD	LDB=	LDM=	STB=	STM=	LDBI	LDMI	STBI	STMI
B	LDBD=	LDMD=	STBD=	STMD=	LDBDX	LDMDX	STBDX	STMDX	LDBI=	LDMI=	STBI=	STMI=	LDBIX	LDMIX	STBIX	STMIX
C	CMB	CMM	ADB	ADM	SBB	SBM	JSBX	ANM	CMB=	CMM=	ADB=	ADM=	SBB=	SBM=	JSB=	ANM=
D	CMBD=	CMMD=	ADBD=	ADMD=	SBBD=	SBMD=	/////	ANMD=	CMBD	CMMD	ADBD	ADMD	SBBD	SBMD	/////	ANMD
E	POBD+	POMD+	POBD-	POMD-	PUBD+	PUMD+	PUBD-	PUMD-	POBI+	POMI+	POBI-	POMI-	PUBI+	PUMI+	PUBI-	PUMI-
F	JMP	JNO	JOD	JEV	JNG	JPS	JNZ	JZR	JEN	JEZ	JNC	JCY	JLZ	JLN	JRZ	JRN

A,01 and D,01 load AR and DR from R00

Figure 5
(a) shows the bit level coding of 75C instructions.
(b) assists in converting from hex opcodes to mnemonics.

Machine code for the 75C is written with a "reverse Polish" style. For example, a sequence to load from R10 into R30 points DRP at R30, ARP at R10, then executes the load operation. A complete description of the intricacies of the 75C instruction set are beyond the scope of this manual. For further information contact the publisher of this manual.

The 75C cpu has the ability to directly address 64K bytes of external memory. This capacity is augmented by bank switching 8K blocks into locations 6000-7FFF. Two such blocks are inside the standard 75C computer. Additional blocks can be plugged into the ports located on the front edge of the computer. The blocks are switched into the 6000-7fff address space by addressing certain bytes in the I/O space. The memory scheme, including the I/O space, of the 75C is shown in figure 6.

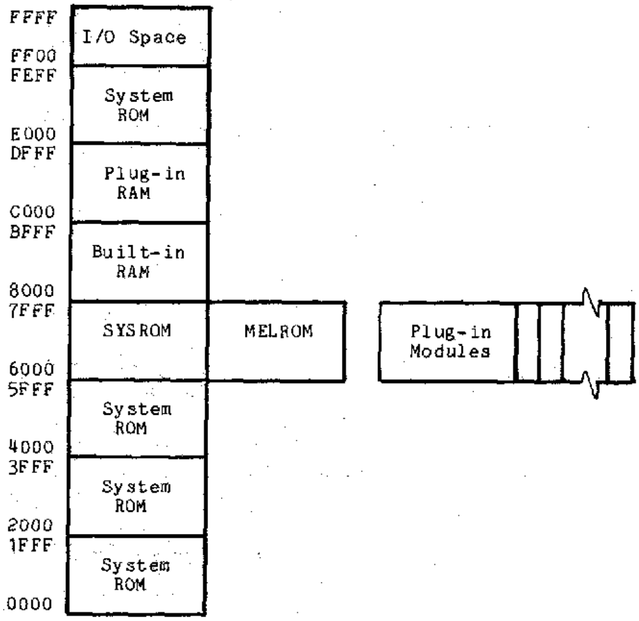


Figure 6
75C memory layout

The top 256 bytes, from FF00 to FFFF are dedicated to I/O. There is no "real" memory at these locations and fetching and storing to these addresses must be done with extreme care. They control, besides ROM switching, LCD display, HP-IL interface, etc. The 75C resident software occupies all of the ROM including two 8K blocks at 6000 and 1354 bytes of RAM from 8000 through 8549 hex. This RAM contains the R06 stack (384 bytes), the character input buffer shared with FORTH (96 bytes) and system variables. It is collectively referred to as the "global area."

User programs all reside in files, each of which has an entry in the directory. The directory starts at 854A and occupies as much space as is required. When more space is needed, as when a new file is added, all the files are shifted upwards and the new entry is added to the directory. Directory entries can be listed from FORTH by typing DIR. command. Files name devfile, iofile and calcprog are used by the operating system.

FORTH ASSEMBLER

The assembler that is precompiled in versions F10L16 and F10V16 is not the whole assembler. If you examine screens 31 and 32 you will see that many of the opcodes are "commented-out"; that is, bracketed with parenthesis so they will not compile. This is to save space. Also, some instructions are used so rarely that they hardly justify the memory space. Another point to note: many condition flags are ignored. These may be easily added if needed.

The FORTH 75C assembler can be run, with some modification, on any FORTH system on any computer. Basically, the assembler insets one byte after another of machine code operator or operand into the dictionary or other designated RAM area. It handles the following control structures: IF . . . THEN; IF . . . ELSE . . . THEN; BEGIN . . . UNTIL; BEGIN . . . AGAIN; and BEGIN . . . WHILE . . . REPEAT. IF, UNTIL and WHILE are conditional and must be preceded by 0= POS or CS, which stand for zero, positive and carry set. Each of these may be turned into its complement by NOT. Thus, POS NOT IF will cause the clause following IF to be executed on negative.

The assembler uses reverse Polish notation. All arguments must be on the stack before the opcode can be processed. The six multibyte literal mode instructions (LDM= STM= CMM= ADM= SBM= ANM=) require a varying number of bytes of operand depending on the value of DRP. The opcode will load, store, etc., the registers from DRP to the next logical boundary with the bytes following the opcode. Thus, during assembly, when the opcode is processed, the correct number of bytes must be on the stack, one byte per stack level. For example,

CODE T1 D,50 7 6 5 4 3 2 1 0 LDM= RTN C ;

will generate machine code 68 A9 00 01 02 03 04 05 06 07 9E and, when executed, registers R50/R57 will be loaded with bytes 0 through 9. The rule of one byte per stack level also applies to the other multibyte instructions. Thus, when constants are used to name addresses the sixteen bit address must be split into two stack levels. The word 1>2 performs this function. For example

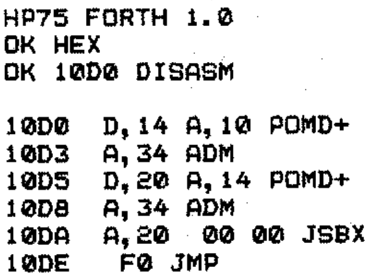
CODE T2 SPARO 1>2 JSB= RTN C ;

will generate machine code C9 84 82 9E where the value of the constant SPARO is 8284, an address where you will find, simply, a RTN instruction.

Although the assembler is designed for the programmer working in hex, it may easily be rewritten for octal. When working with hex, the registers must retain their octal names. In order to integrate the octal registers with the hex, register designations are in the form of A,nn or D,nn where nn is the octal register designation. Note that there is no space between the comma and the digits. This technique takes advantage of the fact that the word headers include both a length byte and a flag marking the end of the word. After A, and D, are compiled, their name length is changed from two to four. When the search routine attempts to match a register, e.g. D,55 with a name in the dictionary, it encounters D, and finds that the lengths are both four bytes, and then compares D and then comma. It stops because it detects the end of name flag, returning with a successful match. A, or D, is then executed, fetching the 55, converting it to the appropriate hex machine code (6D) and inserting it into the dictionary. If this assembler is used on another FORTH, another scheme may be necessary. Examples of FORTH 75C assembler programs are found in the source screens in this manual.

FORTH DISASSEMBLER

There are many parallels between the assembler and disassembler. Among others, it generates reverse Polish code. Disassembly is commenced by placing an address on the stack and executing the word DISASM. Then, with each depression of a key on the keyboard, a line of code is displayed. It assumes that everything is machine code. It cannot distinguish ASCII text. This is a simple tool. With intelligent use it will greatly simplify the task of decyphering code. Figure 7 is an example.



```
HP75 FORTH 1.0
OK HEX
OK 10D0 DISASM

10D0  D,14 A,10 POMD+
10D3  A,34 ADM
10D5  D,20 A,14 POMD+
10D8  A,34 ADM
10DA  A,20  00 00 JSBX
10DE   F0 JMP
```

Figure 7
Example of FORTH disassembler output

1

```
0000                    LST
0000                    EIF
0000 71 00 26           NAM 200,FORTH
0003 00 30 00
0006 2A 00 36
0009 00 3D 00
000C 00 00 9C
000F 10 00 00
0012 3E 00 00
0015 00 00 00
0018 00 00
001A            TWO.    EQU 2
001A            FOUR.   EQU 4
001A            SIX.    EQU 6
001A            !
001A            ! 75FORTH is fig-FORTH for the HP75
001A            !
001A            ! Developed by Dave Conklin
001A            ! With lots of help from Larry Woestman
001A            !   and John Cassady
001A            ! 75FORTH is an adaptation of the Series 80 fig-FORTH
001A            ! developed by Larry Woestman and Tom Houser.
001A            ! Series 80 fig-FORTH is an adaptation of the
001A            ! PDP-11 fig-FORTH developed by the
001A            !      FORTH Interest Group (fig)
001A            !      P.O. Box 1105
001A            !      San Carlos, CA 94070
001A            !
001A            ! REGISTER DESIGNATION = REGISTER ASSIGNMENT
001A            ! IP =  R10,11      FORTH INSTRUCTION POINTER
001A            ! S  =  R32,33      FORTH COMPUTATION STACK POINTER
001A            ! RP =  R0,1        FORTH RETURN-STACK POINTER
001A            ! W  =  R14,R15     FORTH TEMPORARY
001A            !       R34,35      USER AREA RELOCATION CONSTANT
001A            !       R36,R37     FORTH RELOCATION CONSTANT
001A            ! OTHERS USED FOR SCRATCH (EXCEPT R4-7,R12-13,R16-17)
001A            !
```

This listing was generated by an assembler that only knows octal. The four left columns in hex have been dubbed in manually.

2

```
                !       Binary Program Shell
001A            MYBPGM#  EQU 200
001A 71 00               BYT 161,0                  ! LEXID=113 decimal
001C 26 00               DEF RUNTIM
001E 30 00               DEF ASCIIS
0020 2A 00               DEF PARSE
0022 36 00               DEF ERMSG
0024 3D 00               DEF INIT
0026 00 00      RUNTIM   BYT 0,0
0028 9C 10               DEF FORTH.
002A 00 00      PARSE    BYT 0,0
002C 3E 00               DEF FORTH#
002E FF FF               BYT 377,377
0030 46 4F 52   ASCIIS   ASP "FORTH"
0033 54 C8
0035 FF                  BYT 377
0036            ERMSG    BSZ 0
0036 C8                  BYT 200D
0037 4E 4F 4E            ASP "NONE"
003A C5
003B FF                  BYT 377
003C 17                  BYT 027                    ! lex file attributes for HP75
003D 9E         INIT     RTN
003E            FORTH#   BSZ 0
003E 42 B1 A3            LDMD R2,=ROMPTR
0041 82
0042 02 C6 4F            JSB X2,PRSLEX
0045 00
0046 CE 9A 0C            JSB =SFSCAN
0049 6C 06 E3            POMD R54,-R6
004C 0A E5               PUMD R54,+R12
004E 9E                  RTN
004F 42 06 E3   PRSLEX   POMD R2,-R6
0052 6D A8 B4            LDB R55,=EROMTK
0055 6E 21 A1            LDM R56,R41
0058 6D 06 E5            PUMD R55,+R6
005B 4C E4               PUBD R14,+R6
005D 42 E5               PUMD R2,+R6
005F 9E                  RTN
0060            !
```

3

```
                !         origin and initial values for user variables
0060 A1                 BYT 241                 ! BASIC COMMAND ATTRIBUTES
0061            GFORTH  BSZ 0                   ! GLOBAL LABEL
0061 00 00 00   ORIGIN  BSZ 4                   ! COLD START ENTRY POINT
0064 00
0065 00 00 00           BSZ 4
0068 00
0069 00         FRSTAD  BSZ 1                   ! ORIGIN+8
006A            ! ONE BYTE IS MISSING FROM THE CPU # TO ADJUST FOR THE
006A            ! ATTRIBUTES ON WFORTH
006A 4B                 BYT 113                 ! HP75
006B 01 00              BYT 1,0                 ! REV 0.1
006D A1 1A      OR+14   DEF TSK-10              ! POINTER TO LATEST WORD DEFIN
006F 08 00              BYT 10,0                ! BACKSPACE KEY
0071 60 1B      OR+20   DEF XUP                 ! POINTER TO USER AREA
0073 36 28      OR+22   DEF XSO                 ! POINTER TO BEGINNING OF
0075            !                                   STACK
0075 60 1B      OR+24   DEF XRO                 ! POINTER TO BEGINNING OF
0077            !                                   RETURN STACK
0077 CE 1A              DEF XTIB                ! POINTER TO TERMINAL INPUT
0079            !                                   BUFFER
0079 1F 00              BYT 37,0                ! MAXIMUM NAME FIELD WIDTH
007B 00 00              BYT 0,0                 ! WARNING MODE 0=ERROR#,
007D            !                                   1=DISK MESSAGE
007D            !
007D 83         L1      BYT 203                 ! LENGTH 3
007E 4C 49 D4           ASP "LIT"               ! PUSH FOLLOWING ON STACK
0081 00 00              BYT 0,0
0083 85 00      LIT     DEF LIT+
0085 50 08 E1   LIT+    POMD R20,+R10           ! GET LITERAL
0088 1A E7              PUMD R20,-R32           ! PUSH ON STACK
008A 9E                 RTN
008B 87         L2      BYT 207
008C 45 58 45           ASP "EXECUTE"           ! EXECUTE FORTH WORD WHOSE
008F 43 55 54
0092 C5
0093 7D 00              DEF L1                  ! ADDRESS IS ON STACK
0095 97 00      EXEC    DEF EXEC+
0097 50 1A E1   EXEC+   POMD R20,+R32
009A 1C C3              ADM R20,R34
009C 4C 10 A1           LDM R14,R20
009F 50 0C E1           POMD R20,+R14
00A2 1C C3              ADM R20,R34
00A4 8B                 DCM R20
00A5 44 10 A1           LDM R4,R20
00A8            !
```

4

```
00A8 86         L3      BYT 206
00A9 42 52 41           ASP "BRANCH"            ! FORTH BRANCH TO ADDRESS
00AC 4E 43 C8
00AF 8B 00              DEF L2                  ! WHICH FOLLOWS
00B1 B3 00      BRAN    DEF BRAN+
00B3 42 B0 84   BRAN+   LDBD R2,=SVCWRD         ! has a key been hit?
00B6 82
00B7 F2 07              JOD BRAN10              ! yes
00B9 50 08 A5   BRAN05  LDMD R20,R10            ! do the branch
00BC 48 10 C3           ADM R10,R20
00BF 9E                 RTN
00C0            !
00C0 42 B0 5F   BRAN10  LDBD R2,=KEYHIT         ! see if it's the ATTN key
00C3 83
00C4 C8 80              CMB R2,=ATTNKY
00C6 F6 F1              JNZ BRAN05              ! nope
00C8 CE A3 07           JSB =DEQUE              ! get rid of the keyhit
00CB 50 A9 6F           LDM R20,=AB+2           ! yep.  Restart...
00CE 10
00CF 1C C3              ADM R20,R34
00D1 08 A3              STM R20,R10
00D3 9E                 RTN
00D4 87         L4      BYT 207
00D5 30 42 52           ASP "OBRANCH"           ! FORTH BRANCH IF TOP OF
00D8 41 4E 43
00DB C8
00DC A8 00              DEF L3                  ! STACK IS ZERO
00DE E0 00      ZBRAN   DEF ZBRAN+
00E0 50 1A E1   ZBRAN+  POMD R20,+R32
00E3 F7 CE              JZR BRAN+
00E5 48 CB 02   L43$    ADM R10,=2,0            ! SKIP OVER OFFSET
00E8 00
00E9 9E                 RTN
00EA            !
00EA 86         L5      BYT 206
00EB 28 4C 4F           ASP "(LOOP)"            ! INCREMENT LOOP INDEX BY 1,
00EE 4F 50 A9
00F1 D4 00              DEF L4                  ! BRANCH IF BELOW LIMIT
00F3 F5 00      XLOOP   DEF XLOOP+
00F5 50 00 A5   XLOOP+  LDMD R20,R0
00F8 89                 ICM R20
00F9 E5                 PUMD R20,+R0            ! R0 OFF BY +2
00FA D9                 CMMD R20,R0
00FB F5 05              JPS L51$
00FD 40 8B              DCM R0                  ! AIM IT BACK AGAIN
00FF 8B                 DCM R0
0100 F0 B1              JMP BRAN+
0102 40 CB 02   L51$    ADM R0,=2,0             ! CLEAN OFF RETURN STACK
0105 00
0106 F0 DD              JMP L43$
0108            !
```

```
                                                          5
                !        (+LOOP)
0108 87         L6      BYT 207
0109 28 2B 4C           ASP "(+LOOP)"         ! INCREMENT LOOP INDEX BY TOP
010C 4F 4F 50
010F A9
0110 EA 00              DEF L5                ! OF STACK, MAYBE BRANCH
0112 14 01      XPLOO   DEF XPLOO+
0114 50 1A A5   XPLOO+  LDMD R20,R32          ! ADD INDEX TO TOP OF STACK
0117 00 DB              ADMD R20,RO
0119 A7                 STMD R20,RO
011A 1A E1              POMD R20,+R32
011C F4 17              JNG L62$
011E 00 A5              LDMD R20,RO
0120 52 B5 02           LDMD R22,XO,TWO.
0123 00
0124 10 C1              CMM R22,R20
0126 F4 04              JNG L61$
0128 F7 02              JZR L61$
012A F0 87      L6B     JMP BRAN+
012C 40 CB 04   L61$    ADM RO,=4,0
012F 00
0130 48 CB 02           ADM R10,=2,0
0133 00
0134 9E                 RTN
0135 50 00 A5   L62$    LDMD R20,RO           ! HANDLE NEGATIVE INCREMENT
0138 52 B5 02           LDMD R22,XO,TWO.
013B 00
013C 50 12 C1           CMM R20,R22
013F F4 EB              JNG L61$
0141 F7 E9              JZR L61$
0143 F0 E5              JMP L6B
0145 84         L7      BYT 204
0146 28 44 4F           ASP "(DO)"            ! SET UP 'DO' LIMIT AND INDEX
0149 A9
014A 08 01              DEF L6
014C 4E 01      XDO     DEF XDO+
014E 50 1A B5   XDO+    LDMD R20,X32,TWO.
0151 02 00
0153 00 E7              PUMD R20,-RO
0155 1A A5              LDMD R20,R32
0157 00 E7              PUMD R20,-RO
0159 5A CB 04           ADM R32,=4,0
015C 00
015D 9E                 RTN
015E            !
015E 81         L8      BYT 201
015F C9                 ASP "I"               ! RETURN CURRENT LOOP INDEX
0160 45 01              DEF L7                ! TO STACK
0162 64 01      I       DEF I+
0164 50 00 A5   I+      LDMD R20,RO
0167 1A E7              PUMD R20,-R32
0169 9E                 RTN
016A            !
```

```
                                                          6
                !        DIGIT
016A 85         L9      BYT 205
016B 44 49 47           ASP "DIGIT"           ! ASCII-DIGIT BASE => DIGIT-
016E 49 D4
0170 5E 01              DEF L8                ! VALUE TRUE-OR-FALSE
0172 74 01      DIGIT   DEF DIGIT+
0174 50 1A B5   DIGIT+  LDMD R20,X32,TWO.
0177 02 00
0179 CD 30 00           SBM R20,=60,0         ! VALID DIGIT IS ASCII 60 -
017C C9 09 00           CMM R20,=11,0         ! IF GREATER THAN 9,
017F F4 0A              JNG L91$
0181 F7 08              JZR L91$
0183 CD 07 00           SBM R20,=7,0          ! AND THEN IF < 10
0186 C9 0A 00           CMM R20,=12,0
0189 F4 15              JNG L92$              ! ERROR
018B 50 91      L91$    TSM R20
018D F4 11              JNG L92$              ! IF LESS THAN ZERO, ERROR
018F 52 1A A5           LDMD R22,R32
0192 50 12 C1           CMM R20,R22           ! OR IF NOT LESS THAN BASE, ER
0195 F5 09              JPS L92$
0197 1A B7 02           STMD R20,X32,TWO.
019A 00
019B A9 01 00           LDM R20,=1,0
019E A7                 STMD R20,R32          ! VALID RETURN
019F 9E                 RTN
01A0 5A CB 02   L92$    ADM R32,=2,0          ! ERROR, RETURN A 0 FLAG
01A3 00
01A4 50 93              CLM R20
01A6 1A A7              STMD R20,R32
01A8 9E                 RTN
01A9            !
```

```
                                                          7
                !        (FIND)
01A9 86         L10     BYT 206
01AA 28 46 49           ASP "(FIND)"          ! FIND A WORD IN THE
01AD 4E 44 A9
01B0 6A 01              DEF L9                ! DICTIONARY
01B2 B4 01      PFIND   DEF PFIND+
01B4 50 1A E1   PFIND+  POMD R20,+R32         ! GET NFA
01B7 1C C3              ADM R20,R34
01B9 54 10 A1           LDM R24,R20
01BC 50 1A E1           POMD R20,+R32         ! GET STA
01BF 1C C3              ADM R20,R34
01C1 56 10 A1           LDM R26,R20
01C4 50 14 A1           LDM R20,R24
01C7 54 16 A4           LDBD R24,R26          ! GET STRING LENGTH
01CA 58 16 A1   FAST    LDM R30,R26           ! COPY OF STRING POINTER
01CD 52 10 A4           LDBD R22,R20          ! GET NAME LENGTH
01D0 26 A2              STB R22,R46           ! SAVE FOR LATER
01D2 CF 3F FF           ANM R22,=77,377       ! CLEAR FLAG BITS
01D5 14 C0              CMB R22,R24           ! ARE LENGTHS THE SAME
01D7 F7 11              JZR NOFAST            ! JIF YES
01D9 55 10 E0           POBD R25,+R20         ! SKIP LENGTH
01DC 55 10 E0   XMATCH  POBD R25,+R20         ! GET TO END OF STRING
01DF F5 FB              JPS XMATCH
01E1 50 10 A5   XMTCH+  LDMD R20,R20          ! FOLLOW LINK
01E4 F7 30              JZR FAILED            ! JIF AT END OF DIRECTORY
01E6 1C C3              ADM R20,R34
01E8 F0 E0              JMP FAST              ! BACK FOR NEXT NAME
01EA 53 18 E0   NOFAST  POBD R23,+R30         ! SKIP LENGTH
01ED 55 10 E0           POBD R25,+R20         ! SKIP LENGTH
01F0 53 18 E0   MLOOP   POBD R23,+R30         ! GET STRING CHAR
01F3 55 10 E0           POBD R25,+R20         ! GET NAME CHAR
01F6 F4 06              JNG LCHAR
01F8 13 C0              CMB R25,R23           ! ARE THEY THE SAME
01FA F7 F4              JZR MLOOP             ! JIF YES
01FC F0 DE              JMP XMATCH            ! IF NOT, ON TO NEXT NAME
01FE 55 CF 7F   LCHAR   ANM R25,=177          ! GET RID OF TOP BIT
0201 13 C1              CMM R25,R23           ! ARE THEY THE SAME
0203 F6 DC              JNZ XMTCH+            ! JIF NO
0205 50 CB 04           ADM R20,=4,0
0208 00
0209 1C C5              SBM R20,R34
020B 1A E7              PUMD R20,-R32         ! PUT PFA ON STACK
020D 67 92              CLB R47
020F 66 E7              PUMD R46,-R32         ! PUT LENGTH ON STACK
0211 52 93              CLM R22
0213 89                 ICM R22
0214 E7                 PUMD R22,-R32         ! PUT 'TRUE' ON STACK
0215 9E                 RTN
0216 50 93      FAILED  CLM R20
0218 1A E7              PUMD R20,-R32         ! PUT 'FALSE' ON STACK
021A 9E                 RTN
021B            !
```

```
                                                          8
                !        ENCLOSE
021B 87         L11     BYT 207
021C 45 4E 43           ASP "ENCLOSE"         ! BREAK NEXT WORD OUT OF
021F 4C 4F 53
0222 C5
0223 A9 01              DEF L10               ! INPUT BUFFER
0225 27 02      ENCL    DEF ENCL+
0227 54 1A A5   ENCL+   LDMD R24,R32          ! DELIMITER
022A 50 B5 02           LDMD R20,X32,TWO.     ! STARTING ADDRESS
022D 00
022E 1C C3              ADM R20,R34
0230 56 10 A1           LDM R26,R20
0233 5A CD 04           SBM R32,=4,0          ! MAKE SPACE FOR RESULTS
0236 00
0237 50 16 E0   ENC1    POBD R20,+R26
023A 14 C0              CMB R20,R24
023C F7 F9              JZR ENC1              ! SKIP OVER LEADING DELIMITIER
023E 56 8B              DCM R26
0240 1A B7 04           STMD R26,X32,FOUR.
0243 00
0244 50 16 A4   ENC2    LDBD R20,R26
0247 F7 2F              JZR ENC4              ! CHECK FOR NULL
0249 E0                 POBD R20,+R26
024A 14 C0              CMB R20,R24           ! NOT NULL, SO FIND END OF TOK
024C F6 F6              JNZ ENC2
024E 56 1A A7           STMD R26,R32
0251 8B                 DCM R26
0252 56 1A B7   ENC3    STMD R26,X32,TWO.     ! FINISH UP AND RETURN
0255 02 00
0257 50 B5 06           LDMD R20,X32,SIX.
025A 00
025B 1C C3              ADM R20,R34
025D 16 A3              STM R20,R26
025F 1A A5              LDMD R20,R32
0261 16 C5              SBM R20,R26
0263 1A A7              STMD R20,R32
0265 B5 02 00           LDMD R20,X32,TWO.
0268 16 C5              SBM R20,R26
026A 1A B7 02           STMD R20,X32,TWO.
026D 00
026E B5 04 00           LDMD R20,X32,FOUR.
0271 16 C5              SBM R20,R26
0273 1A B7 04           STMD R20,X32,FOUR.
0276 00
0277 9E                 RTN
0278 56 1A A7   ENC4    STMD R26,R32          ! HANDLE NULL CASE
027B 50 B5 04           LDMD R20,X32,FOUR.
027E 00
027F 56 10 C1           CMM R26,R20
0282 F6 CE              JNZ ENC3
0284 89                 ICM R26
0285 F0 CB              JMP ENC3
0287            !
```

```
                                                                  9
                  !          EMIT EMITO KEY KEYO ?TERMINAL CR CRO CMOVE
0287 84          L12       BYT 204
0288 45 4D 49              ASP "EMIT"               ! EMIT KEY
028B D4
028C 1B 02                 DEF L11
028E B8 05       EMIT      DEF DOCOL                ! : EMIT 'EMIT @ EXECUTE ;
0290 46 07                 DEF TEMIT
0292 5D 05                 DEF AT
0294 95 00                 DEF EXEC
0296 12 04                 DEF SEMIS
0298 85          L12.1     BYT 205
0299 45 4D 49              ASP "EMITO"
029C 54 B0
029E 87 02                 DEF L12
02A0 BB 17       EMITO     DEF PEMIT
02A2 84          L13       BYT 204
02A3 4B 45 59              ASP "KEYO"               ! GET A KEY
02A6 B0
02A7 98 02                 DEF L12.1
02A9 DF 17       KEYO      DEF PKEY
02AB             ! PKEY IS NEAR END
02AB 83          L13.1     BYT 203
02AC 4B 45 D9              ASP "KEY"
02AF A2 02                 DEF L13
02B1 B8 05       KEY       DEF DOCOL                ! : KEY 'KEY @ EXECUTE ;
02B3 E3 06                 DEF TKEY
02B5 5D 05                 DEF AT
02B7 95 00                 DEF EXEC
02B9 12 04                 DEF SEMIS
02BB 89          L14       BYT 211
02BC 3F 54 45              ASP "?TERMINAL"          ! BREAK KEY DOWN?
02BF 52 4D 49
02C2 4E 41 CC
02C5 AB 02                 DEF L13.1
02C7 F3 17       QTERM     DEF PQTER
02C9             ! PQTER IS NEAR END
02C9 83          L15       BYT 203
02CA 43 52 B0              ASP "CRO"                ! CR-LF
02CD BB 02                 DEF L14
02CF D0 17       CRO       DEF PCR
02D1             ! PCR IS NEAR END
02D1 82          L15.1     BYT 202
02D2 43 D2                 ASP "CR"
02D4 C9 02                 DEF L15
02D6 B8 05       CR        DEF DOCOL                ! : CR 'CR @ EXECUTE ;
02D8 50 07                 DEF TCR
02DA 5D 05                 DEF AT
02DC 95 00                 DEF EXEC
02DE 12 04                 DEF SEMIS
02E0             !
```

```
                                                                  10
02E0 85          L16       BYT 205
02E1 43 4D 4F              ASP "CMOVE"              ! MOVE BYTES IN MEMORY
02E4 56 C5
02E6 D1 02                 DEF L15.1                ! ( FROM TO COUNT => )
02E8 EA 02       CMOVE     DEF CMOVE+
02EA 52 1A E1 CMOVE+       POMD R22,+R32
02ED F7 15                 JZR L162$                ! NO MOVE
02EF 50 E1                 POMD R20,+R32
02F1 1C C3                 ADM R20,R34
02F3 54 10 A1              LDM R24,R20
02F6 50 1A E1              POMD R20,+R32
02F9 1C C3                 ADM R20,R34
02FB 56 10 E0 L161$        POBD R26,+R20
02FE 14 E4                 PUBD R26,+R24
0300 52 8B                 DCM R22
0302 F6 F7                 JNZ L161$
0304 9E          L162$     RTN
0305 82          L17       BYT 202
0306 55 AA                 ASP "U*"                 ! N1 N2 => 32-BIT PRODUCT
0308 E0 02                 DEF L16                  ! HIGH WORD TOP OF STACK
030A 0C 03       USTAR     DEF UMULT
030C 58 1A E1 UMULT        POMD R30,+R32
030F 50 A9 10              LDM R20,=20,0
0312 00
0313 00 E7                 PUMD R20,-R0
0315 54 93                 CLM R24
0317 56 93                 CLM R26
0319 56 81       LI72$     ELM R26
031B 54 81                 ELM R24
031D 58 81                 ELM R30
031F FA 07                 JNC L171$
0321 56 1A DB              ADMD R26,R32
0324 FA 02                 JNC L171$
0326 54 89                 ICM R24
0328 50 00 A5 L171$        LDMD R20,R0
032B 8B                    DCM R20
032C A7                    STMD R20,R0
032D F6 EA                 JNZ LI72$
032F 56 1A A7              STMD R26,R32
0332 54 E7                 PUMD R24,-R32
0334 50 00 E1              POMD R20,+R0
0337 9E                    RTN
0338             !
```

```
                                                                  11
                  !           U/ AND OR
0338 82          L18       BYT 202
0339 55 AF                 ASP "U/"                 ! UNSIGNED DIVIDE
033B 05 03                 DEF L17
033D 3F 03       USLAS     DEF UDIV
033F 58 1A E1 UDIV         POMD R30,+R32
0342 54 E1                 POMD R24,+R32
0344 56 E1                 POMD R26,+R32
0346 50 A9 10              LDM R20,=20,0
0349 00
034A 56 85       L181$     LLM R26
034C 54 81                 ELM R24
034E F7 09                 JZR L182$
0350 18 C5                 SBM R24,R30
0352 FB 03                 JCY L183$
0354 C3                    ADM R24,R30
0355 F0 02                 JMP L182$
0357 56 89       L183$     ICM R26
0359 50 8B       L182$     DCM R20
035B F6 ED                 JNZ L181$
035D 54 1A E7              PUMD R24,-R32
0360 56 E7                 PUMD R26,-R32
0362 9E                    RTN
0363 83          L19       BYT 203
0364 41 4E C4              ASP "AND"                ! BITWISE AND
0367 38 03                 DEF L18
0369 6B 03       AND       DEF AND+
036B 50 1A E1 AND+         POMD R20,+R32
036E 52 A5                 LDMD R22,R32
0370 10 C7                 ANM R22,R20
0372 1A A7                 STMD R22,R32
0374 9E                    RTN
0375             !
0375 82          L20       BYT 202
0376 4F D2                 ASP "OR"                 ! BITWISE OR
0378 63 03                 DEF L19
037A 7C 03       OR        DEF OR+
037C 50 1A E1 OR+          POMD R20,+R32
037F 52 A5                 LDMD R22,R32
0381 10 95                 ORM R22,R20
0383 1A A7                 STMD R22,R32
0385 9E                    RTN
0386             !
```

```
                                                                  12
                  !           XOR << >> A>> COM
0386 83          L21       BYT 203
0387 58 4F D2              ASP "XOR"                ! BITWISE EXCLUSIVE OR
038A 75 03                 DEF L20
038C 8E 03       XOR       DEF XOR+
038E 54 1A E1 XOR+         POMD R24,+R32
0391 50 A5                 LDMD R20,R32
0393 14 97                 XRM R20,R24
0395 1A A7                 STMD R20,R32
0397 9E                    RTN
0398             !
0398 82          L21.1     BYT 202
0399 3C BC                 ASP "<<"                 ! LOGICAL LEFT SHIFT
039B 86 03                 DEF L21
039D 9F 03       LLS       DEF LLS+
039F 50 1A A5 LLS+         LDMD R20,R32
03A2 85                    LLM R20
03A3 A7                    STMD R20,R32
03A4 9E                    RTN
03A5             !
03A5 82          L21.2     BYT 202
03A6 3E BE                 ASP ">>"                 ! LOGICAL RIGHT SHIFT
03A8 98 03                 DEF L21.1
03AA AC 03       LRS       DEF LRS+
03AC 50 1A A5 LRS+         LDMD R20,R32
03AF 51 87                 LRM R21
03B1 50 A7                 STMD R20,R32
03B3 9E                    RTN
03B4             !
03B4 83          L21.3     BYT 203
03B5 41 3E BE              ASP "A>>"                ! ARITHMETIC RIGHT SHIFT
03B8 A5 03                 DEF L21.2
03BA BC 03       ARS       DEF ARS+
03BC 50 1A A5 ARS+         LDMD R20,R32
03BF 12 A3                 STM R20,R22
03C1 CF 00 80              ANM R20,=0,200           ! GET SIGN
03C4 53 87                 LRM R23                  ! LOGICAL SHIFT
03C6 50 95                 ORM R20,R22              ! PUT SIGN BACK ON
03C8 1A A7                 STMD R20,R32
03CA 9E                    RTN
03CB             !
03CB 83          L21.4     BYT 203
03CC 43 4F CD              ASP "COM"                ! ONE'S COMPLEMENT
03CF B4 03                 DEF L21.3
03D1 D3 03       COM       DEF COM+
03D3 50 1A A5 COM+         LDMD R20,R32
03D6 8F                    NCM R20
03D7 A7                    STMD R20,R32
03D8 9E                    RTN
03D9             !
```

13

```
              !             SP@ SP! RP! ;S LEAVE
03D9 83       L22         BYT 203
03DA 53 50 C0             ASP "SP@"
03DD CB 03                DEF L21.4
03DF E1 03    SPAT        DEF SPAT+
03E1 50 1A A1 SPAT+       LDM R20,R32
03E4 1C C5                SBM R20,R34
03E6 1A E7                PUMD R20,-R32
03E8 9E                   RTN
03E9          !
03E9 83       L23         BYT 203
03EA 53 50 A1             ASP "SP!"
03ED D9 03                DEF L22
03EF F1 03    SPSTO       DEF SPSTO+
03F1 50 A9 36 SPSTO+      LDM R20,=X50
03F4 28
03F5 1C C3                ADM R20,R34
03F7 5A 10 A1             LDM R32,R20
03FA 9E                   RTN
03FB 83       L24         BYT 203
03FC 52 50 A1             ASP "RP!"
03FF E9 03                DEF L23
0401 03 04    RPSTO       DEF RPSTO+
0403 50 A9 60 RPSTO+      LDM R20,=XR0
0406 1B
0407 1C C3                ADM R20,R34
0409 40 10 A1             LDM R0,R20
040C 9E                   RTN
040D 82       L25         BYT 202
040E 3B D3                ASP ";S"
0410 FB 03                DEF L24
0412 14 04    SEMIS       DEF SEMIS+
0414 50 00 E1 SEMIS+      POMD R20,+R0
0417 1C C3                ADM R20,R34
0419 48 10 A1             LDM R10,R20
041C 9E                   RTN
041D 85       L26         BYT 205
041E 4C 45 41             ASP "LEAVE"
0421 56 C5
0423 0D 04                DEF L25
0425 27 04    LEAVE       DEF LEAVE+
0427 50 00 A5 LEAVE+      LDMD R20,R0
042A B7 02 00             STMD R20,X0,TWO.
042D 9E                   RTN
042E          !
```

14

```
              !             >R R> R
042E 82       L27         BYT 202
042F 3E D2                ASP ">R"
0431 1D 04                DEF L26
0433 35 04    TOR         DEF TOR+
0435 50 1A E1 TOR+        POMD R20,+R32
0438 00 E7                PUMD R20,-R0
043A 9E                   RTN
043B          !
043B 82       L28         BYT 202
043C 52 BE                ASP "R>"
043E 2E 04                DEF L27
0440 42 04    FROMR       DEF FROMR+
0442 50 00 E1 FROMR+      POMD R20,+R0
0445 1A E7                PUMD R20,-R32
0447 9E                   RTN
0448          !
0448 81       L29         BYT 201
0449 D2                   ASP "R"
044A 3B 04                DEF L28
044C 4E 04    R           DEF R+
044E 50 00 A5 R+          LDMD R20,R0
0451 1A E7                PUMD R20,-R32
0453 9E                   RTN
0454 82       L30         BYT 202
0455 30 BD                ASP "0="
0457 48 04                DEF L29
0459 5B 04    ZEQU        DEF ZEQU+
045B 50 1A A5 ZEQU+       LDMD R20,R32
045E F7 03                JZR L301$
0460 93                   CLM R20
0461 A7                   STMD R20,R32
0462 9E                   RTN
0463 50 A9 01 L301$       LDM R20,=1,0
0466 00
0467 1A A7                STMD R20,R32
0469 9E                   RTN
046A          !
046A 82       L31         BYT 202
046B 30 BC                ASP "0<"
046D 54 04                DEF L30
046F 71 04    ZLESS       DEF ZLESS+
0471 50 1A A5 ZLESS+      LDMD R20,R32
0474 F4 03                JNG L311$
0476 93                   CLM R20
0477 A7                   STMD R20,R32
0478 9E                   RTN
0479 50 A9 01 L311$       LDM R20,=1,0
047C 00
047D 1A A7                STMD R20,R32
047F 9E                   RTN
0480          !
```

15

```
              !             + D+
0480 81       L32         BYT 201
0481 AB                   ASP "+"
0482 6A 04                DEF L31
0484 86 04    PLUS        DEF PLUS+
0486 50 1A E1 PLUS+       POMD R20,+R32
0489 DB                   ADMD R20,R32
048A A7                   STMD R20,R32
048B 9E                   RTN
048C          !
048C 82       L33         BYT 202
048D 44 AB                ASP "D+"
048F 80 04                DEF L32
0491 93 04    DPLUS       DEF DPLUS+
0493 50 1A B5 DPLUS+      LDMD R20,X32,TWO.
0496 02 00
0498 52 B5 06             LDMD R22,X32,SIX.
049B 00
049C 10 C3                ADM R22,R20
049E FA 09                JNC L331$
04A0 50 1A B5             LDMD R20,X32,FOUR.
04A3 04 00
04A5 89                   ICM R20
04A6 B7 04 00             STMD R20,X32,FOUR.
04A9 50 1A B5 L331$       LDMD R20,X32,FOUR.
04AC 04 00
04AE 52 B7 06             STMD R22,X32,SIX.
04B1 00
04B2 50 DB                ADMD R20,R32
04B4 B7 04 00             STMD R20,X32,FOUR.
04B7 5A CB 04             ADM R32,=4,0
04BA 00
04BB 9E                   RTN
04BC 85       L34         BYT 205
04BD 4D 49 4E             ASP "MINUS"                ! CHANGE SIGN
04C0 55 D3
04C2 8C 04                DEF L33
04C4 C6 04    MINUS       DEF MINUS+
04C6 50 1A A5 MINUS+      LDMD R20,R32
04C9 8D                   TCM R20
04CA A7                   STMD R20,R32
04CB 9E                   RTN
04CC          !
04CC 86       L35         BYT 206
04CD 44 4D 49             ASP "DMINUS"               ! CHS OF DOUBLE INTEGER
04D0 4E 55 D3
04D3 BC 04                DEF L34
04D5 D7 04    DMINU       DEF DMINU+
04D7 50 1A A5 DMINU+      LDMD R20,R32
04DA 8D                   TCM R20
04DB 52 B5 02             LDMD R22,X32,TWO.
04DE 00
04DF 8D                   TCM R22
04E0 FB 02                JCY L351$
04E2 50 8B                DCM R20
04E4 52 1A B7 L351$       STMD R22,X32,TWO.
04E7 02 00
04E9 50 A7                STMD R20,R32
04EB 9E                   RTN
04EC          !
```

16

```
              !             OVER DROP SWAP
04EC 84       L36         BYT 204
04ED 4F 56 45             ASP "OVER"                 ! N1 N2 => N1 N2 N1
04F0 D2
04F1 CC 04                DEF L35
04F3 F5 04    OVER        DEF OVER+
04F5 50 1A B5 OVER+       LDMD R20,X32,TWO.
04F8 02 00
04FA E7                   PUMD R20,-R32
04FB 9E                   RTN
04FC          !
04FC 84       L37         BYT 204
04FD 44 52 4F             ASP "DROP"
0500 D0
0501 EC 04                DEF L36
0503 05 05    DROP        DEF DROP+
0505 5A 89    DROP+       ICM R32
0507 89                   ICM R32
0508 9E                   RTN
0509          !
0509 84       L38         BYT 204
050A 53 57 41             ASP "SWAP"
050D D0
050E FC 04                DEF L37
0510 12 05    SWAP        DEF SWAP+
0512 56 1A B5 SWAP+       LDMD R26,X32,TWO.
0515 02 00
0517 50 A5                LDMD R20,R32
0519 B7 02 00             STMD R20,X32,TWO.
051C 56 A7                STMD R26,R32
051E 9E                   RTN
051F 83       L39         BYT 203
0520 44 55 D0             ASP "DUP"
0523 09 05                DEF L38
0525 27 05    DUP         DEF DUP+
0527 50 1A A5 DUP+        LDMD R20,R32
052A E7                   PUMD R20,-R32
052B 9E                   RTN
052C          !
052C 82       L40         BYT 202
052D 2B A1                ASP "+!"                   ! ADD # SECOND STACK TO ADDRES
052F 1F 05                DEF L39
0531 33 05    PSTOR       DEF PSTOR+
0533 50 1A E1 PSTOR+      POMD R20,+R32              ! GET ADDRESS
0536 1C C3                ADM R20,R34
0538 52 1A E1             POMD R22,+R32              ! GET INCREMENT
053B 10 DB                ADMD R22,R20               ! FORM SUM
053D A7                   STMD R22,R20               ! STORE IT BACK
053E 9E                   RTN
053F          !
```

```
053F 86       L41       BYT 206
0540 54 4F 47           ASP "TOGGLE"            ! BYTE-ADDRESS BIT-PATTERN =>
0543 47 4C C5
0546 2C 05              DEF L40                 ! EXCLUSIVE-OR-INTO-MEMORY-BYT
0548 4A 05    TOGGL     DEF TOGGL+
054A 52 1A E1 TOGGL+    POMD R22,+R32           ! GET BIT PATTERN
054D 50 E1              POMD R20,+R32           ! GET BYTE
054F 1C C3              ADM R20,R34
0551 54 10 A4           LDBD R24,R20
0554 12 96              XRB R24,R22             ! TOGGLE BYTE
0556 10 A6              STBD R24,R20            ! PUT BYTE BACK
0558 9E                 RTN
0559          !
0559 81       L42       BYT 201
055A C0                 ASP "@"
055B 3F 05              DEF L41
055D 5F 05    AT        DEF AT+
055F 50 1A A5 AT+       LDMD R20,R32
0562 1C C3              ADM R20,R34
0564 10 A5              LDMD R20,R20
0566 1A A7              STMD R20,R32
0568 9E                 RTN
0569          !
0569 82       L43       BYT 202
056A 43 C0              ASP "C@"
056C 59 05              DEF L42
056E 70 05    CAT       DEF CAT+
0570 50 1A A5 CAT+      LDMD R20,R32
0573 1C C3              ADM R20,R34
0575 10 A4              LDBD R20,R20
0577 51 92              CLB R21
0579 50 1A A7           STMD R20,R32
057C 9E                 RTN
057D 81       L44       BYT 201
057E A1                 ASP "!"
057F 69 05              DEF L43
0581 83 05    STORE     DEF STORE+
0583 50 1A E1 STORE+    POMD R20,+R32
0586 1C C3              ADM R20,R34
0588 52 1A E1           POMD R22,+R32
058B 10 A7              STMD R22,R20
058D 9E                 RTN
058E          !
058E 82       L45       BYT 202
058F 43 A1              ASP "C!"
0591 7D 05              DEF L44
0593 95 05    CSTOR     DEF CSTOR+
0595 50 1A E1 CSTOR+    POMD R20,+R32
0598 1C C3              ADM R20,R34
059A 52 1A E1           POMD R22,+R32
059D 10 A6              STBD R22,R20
059F 9E                 RTN
05A0          !
```

```
              !               : ; CONSTANT
05A0          !           PRE-COMPILED SECTION
05A0          ! WITH SOME OF THE OPERATIONS
05A0          ! CONVERTED TO CODE FOR SPEED
05A0 C1       L46       BYT 301
05A1 BA                 ASP ":"
05A2 8E 05              DEF L45
05A4 B8 05    COLON     DEF DOCOL
05A6 A7 09              DEF QEXEC
05A8 62 09              DEF SCSP
05AA A0 07              DEF CURR
05AC 5D 05              DEF AT
05AE 92 07              DEF CONT
05B0 81 05              DEF STORE
05B2 4E 0E              DEF CREAT
05B4 2C 0A              DEF RBRAC
05B6 93 0A              DEF PSCOD
05B8 50 08 A1 DOCOL     LDM R20,R10
05BB 1C C5              SBM R20,R34
05BD 00 E7              PUMD R20,-R0
05BF 48 0C A1           LDM R10,R14
05C2 9E                 RTN
05C3          !
05C3 C1       L47       BYT 301
05C4 BB                 ASP ";"
05C5 A0 05              DEF L46
05C7 B8 05    SEMI      DEF DOCOL
05C9 D1 09              DEF QCSP
05CB 08 0A              DEF COMP
05CD 12 04              DEF SEMIS
05CF 41 0A              DEF SMUDG
05D1 1E 0A              DEF LBRAC
05D3 12 04              DEF SEMIS
05D5 88       L48       BYT 210
05D6 43 4F 4E           ASP "CONSTANT"
05D9 53 54 41
05DC 4E D4
05DE C3 05              DEF L47
05E0 B8 05    CON       DEF DOCOL
05E2 4E 0E              DEF CREAT
05E4 41 0A              DEF SMUDG
05E6 2A 08              DEF COMMA
05E8 93 0A              DEF PSCOD
05EA 50 0C A5 DOCON     LDMD R20,R14
05ED 1A E7              PUMD R20,-R32
05EF 9E                 RTN
05F0          !
```

```
              !           VARIABLE USER
05F0 88       L49       BYT 210
05F1 56 41 52           ASP "VARIABLE"
05F4 49 41 42
05F7 4C C5
05F9 D5 05              DEF L48
05FB B8 05    VAR       DEF DOCOL
05FD E0 05              DEF CON
05FF 93 0A              DEF PSCOD
0601 50 0C A1 DOVAR     LDM R20,R14
0604 1C C5              SBM R20,R34
0606 1A E7              PUMD R20,-R32
0608 9E                 RTN
0609 84       L50       BYT 204
060A 55 53 45           ASP "USER"              ! CREATE A NEW USER VARIABLE
060D D2
060E F0 05              DEF L49                 !  (N=>)
0610 B8 05    USER      DEF DOCOL
0612 E0 05              DEF CON
0614 93 0A              DEF PSCOD
0616 50 A9 60 DOUSE     LDM R20,=XUP            ! GET ADDR OF USER VARS
0619 1B
061A 0C DB              ADMD R20,R14            ! ADD USER VAR #
061C 1A E7              PUMD R20,-R32
061E 9E                 RTN
061F          !
061F 81       L51       BYT 201
0620 B0                 ASP "0"
0621 09 06              DEF L50
0623 EA 05    ZERO      DEF DOCON
0625 00 00              BYT 0,0
0627 81       L52       BYT 201
0628 B1                 ASP "1"
0629 1F 06              DEF L51
062B EA 05    ONE       DEF DOCON
062D 01 00              BYT 1,0
062F 81       L53       BYT 201
0630 B2                 ASP "2"
0631 27 06              DEF L52
0633 EA 05    TWO       DEF DOCON
0635 02 00              BYT 2,0
0637 81       L54       BYT 201
0638 B3                 ASP "3"
0639 2F 06              DEF L53
063B EA 05    THREE     DEF DOCON
063D 03 00              BYT 3,0
063F 82       L55       BYT 202
0640 42 CC              ASP "BL"                ! BLANK
0642 37 06              DEF L54
0644 EA 05    BL        DEF DOCON
0646 20 00              BYT 40,0
0648 83       L56       BYT 203
0649 43 2F CC           ASP "C/L"               ! # OF CHARS/LINE
064C 3F 06              DEF L55
064E EA 05    CL        DEF DOCON
0650 20 00              BYT 40,0
0652          !
```

```
0652 85       L57       BYT 205
0653 46 49 52           ASP "FIRST"             ! ADDRESS OF BEGINNING OF DISK
0656 53 D4
0658 48 06              DEF L56
065A EA 05    FIRST     DEF DOCON
065C 38 28              DEF DSKBUF
065E          !
065E 85       L58       BYT 205
065F 4C 49 4D           ASP "LIMIT"             ! ADDRESS JUST BEYOND END OF D
0662 49 D4
0664 52 06              DEF L57
0666 EA 05    LIMIT     DEF DOCON
0668 40 30              DEF ENDBUF
066A 85       L59       BYT 205
066B 42 2F 42           ASP "B/BUF"             ! BYTES PER DISK-BLOCK BUFFER
066E 55 C6
0670 5E 06              DEF L58
0672 EA 05    BBUF      DEF DOCON
0674 00 04              BYT 0,4                 ! 1024 DECIMAL
0676 85       L60       BYT 205
0677 42 2F 53           ASP "B/SCR"             ! DISC BLOCKS PER FORTH SCREEN
067A 43 D2
067C 6A 06              DEF L59
067E EA 05    BSCR      DEF DOCON
0680 01 00              BYT 1,0
0682 82       L62       BYT 202
0683 53 B0              ASP "SO"
0685 76 06              DEF L60
0687 16 06    SZERO     DEF DOUSE
0689 06 00              BYT 6,0
068B 82       L63       BYT 202
068C 52 B0              ASP "RO"                ! RETURN STACK ORIGIN
068E 82 06              DEF L62
0690 16 06    RZERO     DEF DOUSE
0692 08 00              BYT 10,0
0694 83       L64       BYT 203
0695 54 49 C2           ASP "TIB"
0698 8B 06              DEF L63
069A 16 06    TIB       DEF DOUSE
069C 0A 00              BYT 12,0
069E 85       L65       BYT 205
069F 57 49 44           ASP "WIDTH"             ! MAXIMUM NAME LENGTH
06A2 54 C8
06A4 94 06              DEF L64                 ! DEFAULT 31 CHARACTERS
06A6 16 06    WIDTH     DEF DOUSE
06A8 0C 00              BYT 14,0
06AA 87       L66       BYT 207
06AB 57 41 52           ASP "WARNING"           ! WARNING MODE (DEFAULT,
06AE 4E 49 4E
06B1 C7
06B2 9E 06              DEF L65                 ! GIVE MSG # AT ERROR OR WARNI
06B4 16 06    WARN      DEF DOUSE               ! DONT GOTO DISK FOR MSG
06B6 0E 00              BYT 16,0
06B8          !
```

```
                                                                  21
06B8 85            L67       BYT 205
06B9 46 45 4E                ASP "FENCE"             ! PREVENTS FORGET BELOW
06BC 43 C5
06BE AA 06                   DEF L66                 ! THIS FENCE SETTING
06C0 16 06         FENCE     DEF DOUSE
06C2 10 00                   BYT 20,0
06C4 82            L68       BYT 202
06C5 44 D0                   ASP "DP"                ! DICTIONARY PTR TO
06C7 B8 06                   DEF L67
06C9 16 06         DP        DEF DOUSE               ! NEXT AVALBL SPACE
06CB 12 00                   BYT 22,0
06CD 88            L69       BYT 210
06CE 56 4F 43                ASP "VOC-LINK"          ! VOCABULARY LINK
06D1 2D 4C 49
06D4 4E CB
06D6 C4 06                   DEF L68
06D8 16 06         VOCL      DEF DOUSE
06DA 14 00                   BYT 24,0
06DC 84            L69.1     BYT 204
06DD 27 4B 45                ASP "'KEY"              ! KEY vector
06E0 D9
06E1 CD 06                   DEF L69
06E3 16 06         TKEY      DEF DOUSE
06E5 16 00                   BYT 26,0
06E7 86            L69.2     BYT 206
06E8 4F 55 54                ASP "OUTBUF"            ! OUTPUT BUFFER POINTER
06EB 42 55 C6
06EE DC 06                   DEF L69.1
06F0 16 0E         OUTBUF    DEF DOUSE
06F2 18 00                   BYT 30,0
06F4 86            L69.3     BYT 206
06F5 49 4E 50                ASP "INPBUF"            ! INPUT BUFFER POINTER
06F8 42 55 C6
06FB E7 06                   DEF L69.2
06FD 16 06         INPBF     DEF DOUSE
06FF 1A 00                   BYT 32,0
0701 83            L69.4     BYT 203
0702 55 53 C5                ASP "USE"               ! LEAST RECENT DISK BUFFER
0705 F4 06                   DEF L69.3
0707 16 06         USE       DEF DOUSE
0709 1C 00                   BYT 34,0
070B 84            L69.5     BYT 204
070C 50 52 45                ASP "PREV"              ! MOST RECENT DISK BUFFER
070F D6
0710 01 07                   DEF L69.4
0712 16 06         PREV      DEF DOUSE
0714 1E 00                   BYT 36,0
0716               !
```

```
                                                                  22
                   !          OKFLAG 'NUMBER' 'ABORT'
0716 86            L69.6     BYT 206
0717 4F 4B 46                ASP "OKFLAG"            ! ENABLE/DISABLE 'OK' IN QUIT
071A 4C 41 C7
071D 0B 07                   DEF L69.5
071F 16 06         OKFLG     DEF DOUSE
0721 20 00                   BYT 40,0
0723 87            L69.7     BYT 207
0724 27 4E 55                ASP "'NUMBER"           ! ALLOWS REDIRECTION OF NUMBER
0727 4D 42 45
072A D2
072B 16 07                   DEF L69.6
072D 16 06         SNUMB     DEF DOUSE
072F 22 00                   BYT 42,0
0731 86            L69.8     BYT 206
0732 27 41 42                ASP "'ABORT"            ! ALLOWS REDIRECTION OF ABORT
0735 4F 52 D4
0738 23 07                   DEF L69.7
073A 16 06         SABOR     DEF DOUSE
073C 24 00                   BYT 44,0
073E 85            L69.9     BYT 205
073F 27 45 4D                ASP "'EMIT"
0742 49 D4
0744 31 07                   DEF L69.8
0746 16 06         TEMIT     DEF DOUSE
0748 26 00                   BYT 46,0
074A 83            L69.91    BYT 203
074B 27 43 D2                ASP "'CR"
074E 3E 07                   DEF L69.9
0750 16 06         TCR       DEF DOUSE
0752 28 00                   BYT 50,0
0754 83            L70       BYT 203
0755 42 4C CB                ASP "BLK"               ! CURRENT DISK BLOCK
0758 4A 07                   DEF L69.91              ! BEING LOADED(0=TERMINAL
075A 16 06         BLK       DEF DOUSE
075C 2A 00                   BYT 52,0
075E 82            L71       BYT 202
075F 49 CE                   ASP "IN"                ! OFFSET IN TERMINAL
0761 54 07                   DEF L70                 ! INPUT BUFFER
0763 16 06         IN        DEF DOUSE
0765 2C 00                   BYT 54,0
0767 83            L72       BYT 203
0768 4F 55 D4                ASP "OUT"               ! OFFSET IN OUTPUT LINE
076B 5E 07                   DEF L71
076D 16 06         OUT       DEF DOUSE
076F 2E 00                   BYT 56,0
0771 83            L73       BYT 203
0772 53 43 D2                ASP "SCR"               ! CURRENT FORTH DISK SCREEN
0775 67 07                   DEF L72
0777 16 06         SCR       DEF DOUSE
0779 30 00                   BYT 60,0
077B 86            L74       BYT 206
077C 4F 46 46                ASP "OFFSET"            ! OFFSET TO GET
077F 53 45 D4
0782 71 07                   DEF L73                 ! ANOTHER DISK DRIVE
0784 16 06         OFSET     DEF DOUSE
0786 32 00                   BYT 62,0
0788               !
```

```
                                                                  23
0788 87            L75       BYT 207
0789 43 4F 4E                ASP "CONTEXT"
078C 54 45 58
078F D4
0790 7B 07                   DEF L74
0792 16 06         CONT      DEF DOUSE
0794 34 00                   BYT 64,0
0796 87            L76       BYT 207
0797 43 55 52                ASP "CURRENT"
079A 52 45 4E
079D D4
079E 88 07                   DEF L75
07A0 16 06         CURR      DEF DOUSE
07A2 36 00                   BYT 66,0
07A4 85            L77       BYT 205
07A5 53 54 41                ASP "STATE"
07A8 54 C5
07AA 96 07                   DEF L76
07AC 16 06         STATE     DEF DOUSE
07AE 38 00                   BYT 70,0
07B0 84            L78       BYT 204
07B1 42 41 53                ASP "BASE"
07B4 C5
07B5 A4 07                   DEF L77
07B7 16 06         BASE      DEF DOUSE
07B9 3A 00                   BYT 72,0
07BB 83            L79       BYT 203
07BC 44 50 CC                ASP "DPL"               ! OFFSET OF DEC POINT
07BF B0 07                   DEF L78                 ! AFTER DOUBLE INT INPUT
07C1 16 06         DPL       DEF DOUSE
07C3 3C 00                   BYT 74,0
07C5 83            L80       BYT 203
07C6 46 4C C4                ASP "FLD"               ! OUTPUT FIELD WIDTH
07C9 BB 07                   DEF L79
07CB 16 06         FLD       DEF DOUSE
07CD 3E 00                   BYT 76,0
07CF 83            L81       BYT 203
07D0 43 53 D0                ASP "CSP"               ! USED BY COMPILER TO
07D3 C5 07                   DEF L80                 ! HOLD CUR STK POSITION
07D5 16 06         CSP       DEF DOUSE               ! FOR ERROR CHECKING
07D7 40 00                   BYT 100,0
07D9 82            L82       BYT 202
07DA 52 A3                   ASP "R#"                ! USED BY EDITOR
07DC CF 07                   DEF L81                 ! FOR CURSOR POSITION
07DE 16 06         RNUM      DEF DOUSE
07E0 42 00                   BYT 102,0
07E2 83            L83       BYT 203
07E3 48 4C C4                ASP "HLD"               ! POINTS TO LAST CHAR
07E6 D9 07                   DEF L82                 ! HELD IN PAD
07E8 16 06         HLD       DEF DOUSE
07EA 44 00                   BYT 104,0
07EC               !
```

```
                                                                  24
                   !          1+ 2+ HERE ALLOT
07EC 82            L84.1     BYT 202
07ED 31 AB                   ASP "1+"
07EF E2 07                   DEF L83
07F1 F3 07         ONEP      DEF ONEP+
07F3 50 1A A5 ONEP+          LDMD R20,R32
07F6 89                      ICM R20
07F7 A7                      STMD R20,R32
07F8 9E                      RTN
07F9               !
07F9 82            L85       BYT 202
07FA 32 AB                   ASP "2+"
07FC EC 07                   DEF L84.1
07FE 00 08         TWOP      DEF TWOP+
0800 50 1A A5 TWOP+          LDMD R20,R32
0803 89                      ICM R20
0804 89                      ICM R20
0805 A7                      STMD R20,R32
0806 9E                      RTN
0807               !
0807 84            L86       BYT 204
0808 48 45 52                ASP "HERE"
080B C5
080C F9 07                   DEF L85
080E B8 05         HERE      DEF DOCOL
0810 C9 06                   DEF DP
0812 5D 05                   DEF AT
0814 12 04                   DEF SEMIS
0816 85            L87       BYT 205
0817 41 4C 4C                ASP "ALLOT"
081A 4F D4
081C 07 08                   DEF L86
081E B8 05         ALLOT     DEF DOCOL
0820 C9 06                   DEF DP
0822 31 05                   DEF PSTOR
0824 12 04                   DEF SEMIS
0826 81            L88       BYT 201
0827 AC                      ASP ","                 ! COMMA
0828 16 08                   DEF L87
082A B8 05         COMMA     DEF DOCOL
082C 0E 08                   DEF HERE
082E 81 05                   DEF STORE
0830 33 06                   DEF TWO
0832 1E 08                   DEF ALLOT
0834 12 04                   DEF SEMIS
0836 82            L88.5     BYT 202
0837 43 AC                   ASP "C,"                ! C-COMMA
0839 26 08                   DEF L88
083B B8 05         CCOMA     DEF DOCOL
083D 0E 08                   DEF HERE
083F 93 05                   DEF CSTOR
0841 2B 06                   DEF ONE
0843 1E 08                   DEF ALLOT
0845 12 04                   DEF SEMIS
0847               !
```

```
                                                            25

0847 81       L89     BYT 201
0848 AD               ASP "-"                 ! SUBTRACT
0849 36 08            DEF L88.5
084B 4D 08    SUB     DEF SUB+
084D 50 1A E1 SUB+    POMD R20,+R32
0850 52 A5            LDMD R22,R32
0852 10 C5            SBM R22,R20
0854 1A A7            STMD R22,R32
0856 9E               RTN
0857          !
0857 81       L90     BYT 201
0858 BD               ASP "="
0859 47 08            DEF L89
085B 5D 08    EQUAL   DEF EQUAL+
085D 52 93    EQUAL+  CLM R22
085F 50 1A E1         POMD R20,+R32
0862 D9               CMMD R20,R32
0863 F6 02            JNZ L901
0865 52 89            ICM R22
0867 52 1A A7 L901    STMD R22,R32
086A 9E               RTN
086B 81       L91     BYT 201
086C BC               ASP "<"                 ! SIGNED <
086D 57 08            DEF L90
086F 71 08    LESS    DEF LESS+
0871 52 93    LESS+   CLM R22
0873 89               ICM R22
0874 50 1A E1         POMD R20,+R32
0877 54 A5            LDMD R24,R32
0879 10 C1            CMM R24,R20
087B F4 02            JNG L911
087D 52 8B            DCM R22
087F 52 1A A7 L911    STMD R22,R32
0882 9E               RTN
0883          !
0883 81       L92     BYT 201
0884 BE               ASP ">"                 ! SIGNED >
0885 6B 08            DEF L91
0887 89 08    GREAT   DEF GREAT+
0889 52 93    GREAT+  CLM R22
088B 89               ICM R22
088C 54 1A E1         POMD R24,+R32
088F 50 A5            LDMD R20,R32
0891 54 10 C1         CMM R24,R20
0894 F4 02            JNG L921
0896 52 8B            DCM R22
0898 52 1A A7 L921    STMD R22,R32
089B 9E               RTN
089C          !
```

```
                                                            26
              !         ROT SPACE
089C 83       L93     BYT 203
089D 52 4F D4         ASP "ROT"               ! ROTATE FIRST 3
08A0 83 08            DEF L92
08A2 A4 08    ROT     DEF ROT+
08A4 54 1A A5 ROT+    LDMD R24,R32
08A7 52 B5 04         LDMD R22,X32,FOUR.
08AA 00
08AB A7               STMD R22,R32
08AC B5 02 00         LDMD R22,X32,TWO.
08AF B7 04 00         STMD R22,X32,FOUR.
08B2 54 B7 02         STMD R24,X32,TWO.
08B5 00
08B6 9E               RTN
08B7          !
08B7 85       L94     BYT 205
08B8 53 50 41         ASP "SPACE"             !SPACE
08BB 43 C5
08BD 9C 08            DEF L93
08BF B8 05    SPACE   DEF DOCOL
08C1 83 00            DEF LIT
08C3 20 00            BYT 40,0
08C5 8E 02            DEF EMIT
08C7 12 04            DEF SEMIS
08C9 84       L95     BYT 204
08CA 2D 44 55         ASP "-DUP"
08CD D0
08CE B7 08            DEF L94
08D0 D2 08    DDUP    DEF DDUP+               ! -DUP
08D2 50 1A A5 DDUP+   LDMD R20,R32
08D5 F7 01            JZR L951
08D7 E7               PUMD R20,-R32
08D8 9E       L951    RTN
08D9          !
08D9 88       L96     BYT 210
08DA 54 52 41         ASP "TRAVERSE"          ! MOVE (FORWARDS OR
08DD 56 45 52
08E0 53 C5
08E2 C9 08            DEF L95                 ! BACKWARDS) ACROSS
08E4 B8 05    TRAV    DEF DOCOL               ! A (VAR LEN)
08E6 10 05            DEF SWAP                ! DICITIONARY NAME FIELD
08E8 F3 04    XXN1    DEF OVER
08EA 84 04            DEF PLUS
08EC 83 00            DEF LIT
08EE 7F 00            BYT 177,0
08F0 F3 04            DEF OVER
08F2 6E 05            DEF CAT
08F4 6F 08            DEF LESS
08F6 DE 00            DEF ZBRAN
08F8 F0 FF            BYT 360,377             ! XXN1
08FA 10 05            DEF SWAP
08FC 03 05            DEF DROP
08FE 12 04            DEF SEMIS
0900          !
```

```
                                                            27
              !         LATEST LFA CFA NFA PFA
0900 86       L96+    BYT 206
0901 4C 41 54         ASP "LATEST"
0904 45 53 D4
0907 D9 08            DEF L96
0909 B8 05    LATES   DEF DOCOL
090B A0 07            DEF CURR
090D 5D 05            DEF AT
090F 5D 05            DEF AT
0911 12 04            DEF SEMIS
0913 83       L97     BYT 203
0914 4C 46 C1         ASP "LFA"               ! LFA
0917 00 09            DEF L96+
0919 B8 05    LFA     DEF DOCOL
091B 83 00            DEF LIT
091D 04 00            BYT 4,0
091F 4B 08            DEF SUB
0921 12 04            DEF SEMIS
0923 83       L98     BYT 203
0924 43 46 C1         ASP "CFA"               ! CFA
0927 13 09            DEF L97
0929 B8 05    CFA     DEF DOCOL
092B 33 06            DEF TWO
092D 4B 08            DEF SUB
092F 12 04            DEF SEMIS
0931 83       L99     BYT 203
0932 4E 46 C1         ASP "NFA"               ! NFA
0935 23 09            DEF L98
0937 B8 05    NFA     DEF DOCOL
0939 83 00            DEF LIT
093B 05 00            BYT 5,0
093D 4B 08            DEF SUB
093F 83 00            DEF LIT
0941 FF FF            BYT 377,377
0943 E4 08            DEF TRAV
0945 12 04            DEF SEMIS
0947 83       L100    BYT 203
0948 50 46 C1         ASP "PFA"               ! PFA
094B 31 09            DEF L99
094D B8 05    PFA     DEF DOCOL
094F 2B 06            DEF ONE
0951 E4 08            DEF TRAV
0953 83 00            DEF LIT
0955 05 00            BYT 5,0
0957 84 04            DEF PLUS
0959 12 04            DEF SEMIS
095B          !
```

```
              !         !CSP ?ERROR ?COMP                   28
095B          ! THE NEXT 7 OPERATIONS ARE USED
095B          ! BY THE COMPILER FOR COMPILE
095B          ! TIME SYNTAX-ERROR CHECKS
095B 84       L101    BYT 204
095C 21 43 53         ASP "!CSP"              ! CSP
095F D0
0960 47 09            DEF L100
0962 B8 05    SCSP    DEF DOCOL
0964 DF 03            DEF SPAT
0966 D5 07            DEF CSP
0968 81 05            DEF STORE
096A 12 04            DEF SEMIS
096C 86       L102    BYT 206
096D 3F 45 52         ASP "?ERROR"            ! ?ERROR
0970 52 4F D2
0973 5B 09            DEF L101
0975 B8 05    QERR    DEF DOCOL
0977 10 05            DEF SWAP
0979 DE 00            DEF ZBRAN
097B 08 00            BYT 10,0                ! XXN2
097D E7 0D            DEF ERROR
097F B1 00            DEF BRAN
0981 04 00            BYT 4,0                 ! XXN3
0983 03 05    XXN2    DEF DROP
0985 12 04    XXN3    DEF SEMIS
0987 85       L103    BYT 205
0988 3F 43 4F         ASP "?COMP"             ! ?COMP
098B 4D D0
098D 6C 09            DEF L102
098F B8 05    QCOMP   DEF DOCOL
0991 AC 07            DEF STATE
0993 5D 05            DEF AT
0995 59 04            DEF ZEQU
0997 83 00            DEF LIT
0999 11 00            BYT 21,0
099B 75 09            DEF QERR
099D 12 04            DEF SEMIS
099F 85       L104    BYT 205
09A0 3F 45 58         ASP "?EXEC"
09A3 45 C3
09A5 87 09            DEF L103
09A7 B8 05    QEXEC   DEF DOCOL
09A9 AC 07            DEF STATE
09AB 5D 05            DEF AT
09AD 83 00            DEF LIT
09AF 12 00            BYT 22,0
09B1 75 09            DEF QERR
09B3 12 04            DEF SEMIS
09B5 86       L105    BYT 206
09B6 3F 50 41         ASP "?PAIRS"
09B9 49 52 D3
09BC 9F 09            DEF L104
09BE B8 05    QPAIR   DEF DOCOL
09C0 4B 08            DEF SUB
09C2 83 00            DEF LIT
09C4 13 00            BYT 23,0
09C6 75 09            DEF QERR
09C8 12 04            DEF SEMIS
09CA          !
```

```
                                                            29
                !          ?CSP ?LOADING COMPILE
09CA 84         L106      BYT 204
09CB 3F 43 53             ASP "?CSP"
09CE D0
09CF B5 09                DEF L105
09D1 B8 05      QCSP      DEF DOCOL
09D3 DF 03                DEF SPAT
09D5 D5 07                DEF CSP
09D7 5D 05                DEF AT
09D9 4B 08                DEF SUB
09DB 83 00                DEF LIT
09DD 14 00                BYT 24,0
09DF 75 09                DEF QERR
09E1 12 04                DEF SEMIS
09E3 88         L107      BYT 210
09E4 3F 4C 4F             ASP "?LOADING"
09E7 41 44 49
09EA 4E C7
09EC CA 09                DEF L106
09EE B8 05      QLOAD     DEF DOCOL
09F0 5A 07                DEF BLK
09F2 5D 05                DEF AT
09F4 59 04                DEF ZEQU
09F6 83 00                DEF LIT
09F8 16 00                BYT 26,0
09FA 75 09                DEF QERR
09FC 12 04                DEF SEMIS
09FE 87         L108      BYT 207
09FF 43 4F 4D             ASP "COMPILE"             ! COMPILE THE EXECUTION ADDRES
0A02 50 49 4C
0A05 C5
0A06 E3 09                DEF L107
0A08 B8 05      COMP      DEF DOCOL
0A0A 8F 09                DEF QCOMP
0A0C 40 04                DEF FROMR
0A0E 25 05                DEF DUP
0A10 FE 07                DEF TWOP
0A12 33 04                DEF TOR
0A14 5D 05                DEF AT
0A16 2A 08                DEF COMMA
0A18 12 04                DEF SEMIS
0A1A C1         L109      BYT 301
0A1B DB                   ASP "["                   ! STOP COMPILATION, ENTER EXEC
0A1C FE 09                DEF L108
0A1E B8 05      LBRAC     DEF DOCOL
0A20 23 06                DEF ZERO
0A22 AC 07                DEF STATE
0A24 81 05                DEF STORE
0A26 12 04                DEF SEMIS
0A28 81         L110      BYT 201
0A29 DD                   ASP "]"                   ! ENTER COMPILATION STATE
0A2A 1A 0A                DEF L109
0A2C B8 05      RBRAC     DEF DOCOL
0A2E 83 00                DEF LIT
0A30 C0 00                BYT 300,0
0A32 AC 07                DEF STATE
0A34 81 05                DEF STORE
0A36 12 04                DEF SEMIS
0A38            !
```

```
                                                            30
0A38 86         L111      BYT 206
0A39 53 4D 55             ASP "SMUDGE"              ! ALTER LATEST WORD NAME SO TH
0A3C 44 47 C5
0A3F 28 0A                DEF L110                  ! DICTIONARY SEARCH WON'T FIND
0A41            !                                      PARTIALLY-COMPLETED ENTRY
0A41 B8 05      SMUDG     DEF DOCOL
0A43 09 09                DEF LATES
0A45 83 00                DEF LIT
0A47 20 00                BYT 40,0
0A49 48 05                DEF TOGGL
0A4B 12 04                DEF SEMIS
0A4D 83         L112      BYT 203
0A4E 48 45 D8             ASP "HEX"
0A51 38 0A                DEF L111
0A53 B8 05      HEX       DEF DOCOL
0A55 83 00                DEF LIT
0A57 10 00                BYT 20,0
0A59 B7 07                DEF BASE
0A5B 81 05                DEF STORE
0A5D 12 04                DEF SEMIS
0A5F 87         L113      BYT 207
0A60 44 45 43             ASP "DECIMAL"
0A63 49 4D 41
0A66 CC
0A67 4D 0A                DEF L112
0A69 B8 05      DEC       DEF DOCOL
0A6B 83 00                DEF LIT
0A6D 0A 00                BYT 12,0
0A6F B7 07                DEF BASE
0A71 81 05                DEF STORE
0A73 12 04                DEF SEMIS
0A75 85         L114      BYT 205
0A76 4F 43 54             ASP "OCTAL"
0A79 41 CC
0A7B 5F 0A                DEF L113
0A7D B8 05      OCTAL     DEF DOCOL
0A7F 83 00                DEF LIT
0A81 08 00                BYT 10,0
0A83 B7 07                DEF BASE
0A85 81 05                DEF STORE
0A87 12 04                DEF SEMIS
0A89 87         L115      BYT 207
0A8A 28 3B 43             ASP "(;CODE)"             ! USED ONLY BY COMPILER
0A8D 4F 44 45
0A90 A9
0A91 75 0A                DEF L114                  ! COMPILED BY ';CODE'
0A93 B8 05      PSCOD     DEF DOCOL
0A95 40 04                DEF FROMR
0A97 09 09                DEF LATES
0A99 4D 09                DEF PFA
0A9B 29 09                DEF CFA
0A9D 81 05                DEF STORE
0A9F 12 04                DEF SEMIS
0AA1            !
```

```
                                                            31
0AA1 87         L117      BYT 207
0AA2 3C 42 55             ASP "<BUILDS"             ! CREATE NEW DATA TYPE WITH CO
0AA5 49 4C 44
0AA8 D3
0AA9 89 0A                DEF L115                  ! ROUTINE IN HIGHER-LEVEL FORT
0AAB B8 05      BUILD     DEF DOCOL
0AAD 23 06                DEF ZERO
0AAF E0 05                DEF CON
0AB1 12 04                DEF SEMIS
0AB3 85         L118      BYT 205
0AB4 44 4F 45             ASP "DOES>"
0AB7 53 BE
0AB9 A1 0A                DEF L117
0ABB B8 05      DOES      DEF DOCOL
0ABD 40 04                DEF FROMR
0ABF 09 09                DEF LATES
0AC1 4D 09                DEF PFA
0AC3 81 05                DEF STORE
0AC5 93 0A                DEF PSCOD
0AC7 50 08 A1 DODOE       LDM R20,R10
0ACA 1C C5                SBM R20,R34
0ACC 00 E7                PUMD R20,-R0
0ACE 0C E1                POMD R20,+R14
0AD0 1C C3                ADM R20,R34
0AD2 48 10 A1             LDM R10,R20
0AD5 50 0C A1             LDM R20,R14
0AD8 1C C5                SBM R20,R34
0ADA 1A E7                PUMD R20,-R32
0ADC 9E                   RTN
0ADD 85         L119      BYT 205
0ADE 43 4F 55             ASP "COUNT"               ! CONVERT STRING TO THE FORMAT
0AE1 4E D4
0AE3 B3 0A                DEF L118                  ! USED BY 'TYPE'
0AE5 B8 05      COUNT     DEF DOCOL
0AE7 25 05                DEF DUP
0AE9 F1 07                DEF ONEP
0AEB 10 05                DEF SWAP
0AED 6E 05                DEF CAT
0AEF 12 04                DEF SEMIS
0AF1            !
```

```
                                                            32
0AF1 84         L120      BYT 204
0AF2 54 59 50             ASP "TYPE"
0AF5 C5
0AF6 DD 0A                DEF L119
0AF8 B8 05      TYPE      DEF DOCOL
0AFA D0 08                DEF DDUP
0AFC DE 00                DEF ZBRAN
0AFE 18 00                BYT 30,0                  ! XXL2
0B00 F3 04                DEF OVER
0B02 84 04                DEF PLUS
0B04 10 05                DEF SWAP
0B06 4C 01                DEF XDO
0B08 62 01      XXL1      DEF I
0B0A 6E 05                DEF CAT
0B0C 8E 02                DEF EMIT
0B0E F3 00                DEF XLOOP
0B10 F8 FF                BYT 370,377               !XXL1
0B12 B1 00                DEF BRAN
0B14 04 00                BYT 4,0                   !XXL3
0B16 03 05      XXL2      DEF DROP
0B18 12 04      XXL3      DEF SEMIS
0B1A 89         L122      BYT 211
0B1B 2D 54 52             ASP "-TRAILING"
0B1E 41 49 4C
0B21 49 4E C7
0B24 F1 0A                DEF L120
0B26 B8 05      DTRAI     DEF DOCOL
0B28 25 05                DEF DUP
0B2A 23 06                DEF ZERO
0B2C 4C 01                DEF XDO
0B2E F3 04      XXW6      DEF OVER
0B30 F3 04                DEF OVER
0B32 84 04                DEF PLUS
0B34 2B 06                DEF ONE
0B36 4B 08                DEF SUB
0B38 6E 05                DEF CAT
0B3A 44 06                DEF BL
0B3C 4B 08                DEF SUB
0B3E DE 00                DEF ZBRAN
0B40 08 00                BYT 10,0                  !XXW7
0B42 25 04                DEF LEAVE
0B44 B1 00                DEF BRAN
0B46 06 00                BYT 6,0                   !XXWA
0B48 2B 06      XXW7      DEF ONE
0B4A 4B 08                DEF SUB
0B4C F3 00      XXWA      DEF XLOOP
0B4E E0 FF                BYT 340,377               !XXW6
0B50 12 04                DEF SEMIS
0B52            !
```

```
                                                          33
                !       (.") ."
0B52 84         L123    BYT 204
0B53 28 2E 22           ASP 4,(.")
0B56 A9
0B57 1A 0B              DEF L122                ! USED ONLY BY COMPILER
0B59 B8 05      PDOTQ   DEF DOCOL               ! COMPILED BY '."'
0B5B 4C 04              DEF R
0B5D E5 0A              DEF COUNT
0B5F 25 05              DEF DUP
0B61 F1 07              DEF ONEP
0B63 40 04              DEF FROMR
0B65 84 04              DEF PLUS
0B67 33 04              DEF TOR
0B69 F8 0A              DEF TYPE
0B6B 12 04              DEF SEMIS
0B6D C2         L124    BYT 302
0B6E 2E A2              ASP 2,."
0B70 52 0B              DEF L123                ! TYPE ASCII MESSAGE
0B72 B8 05      DOTQ    DEF DOCOL
0B74 83 00              DEF LIT
0B76 22 00              BYT 42,0                ! ASCII "
0B78 AC 07              DEF STATE
0B7A 5D 05              DEF AT
0B7C DE 00              DEF ZBRAN
0B7E 14 00              BYT 24,0                ! XXL6
0B80 08 0A              DEF COMP
0B82 59 0B              DEF PDOTQ
0B84 9C 0C              DEF WORD
0B86 0E 08              DEF HERE
0B88 6E 05              DEF CAT
0B8A F1 07              DEF ONEP
0B8C 1E 08              DEF ALLOT
0B8E B1 00              DEF BRAN
0B90 0A 00              BYT 12,0                ! XXL7
0B92 9C 0C      XXL6    DEF WORD
0B94 0E 08              DEF HERE
0B96 E5 0A              DEF COUNT
0B98 F8 0A              DEF TYPE
0B9A 12 04      XXL7    DEF SEMIS
0B9C            !
```

```
                                                          34
              !         EXPECT
0B9C          ! EXPECT ( adr count - )
0B9C          ! Gets a line of input from the keyboard and stores it at adr
0B9C          ! followed by 2 nulls.  The CR is not stored.  No more than
0B9C          ! <count> characters (+ 2 nulls) will be stored at adr.
0B9C          !
0B9C 86       L126      BYT 206
0B9D 45 58 50           ASP "EXPECT"
0BA0 45 43 D4
0BA3 6D 0B              DEF L124
0BA5 A7 0B    EXPEC     DEF EXPEC+
0BA7 1E C6 B0 EXPEC+    JSB X36,SAVFVM
0BAA 1A
0BAB          ! If you believe the KR entry point documentation, it may onl
0BAB          ! necessary to save and restore ROm here.
0BAB CE 6D 41           JSB =GET.IN
0BAE          ! GET.IN returns R24=line length, R25=terminating character,
0BAE          ! in INPBUF with CR after last character
0BAE 5E 06 E3           POMD R36,-R6
0BB1 1E C6 C0           JSB X36,GETFVM
0BB4 1A
0BB5          ! now move line from INPBUF to adr
0BB5 55 92              CLB R25                 ! R24m will be compared to cou
0BB7 52 1A E1           POMD R22,+R32           ! get desired count
0BBA 54 12 C1           CMM R24,R22             ! compare to actual count
0BBD F5 01              JPS EXPE10              ! jump if actual>=desired
0BBF A3                 STM R24,R22             ! actual<desired - use actual
0BC0          !
0BC0 50 1A E1 EXPE10    POMD R20,+R32           ! get adr
0BC3 1C C3              ADM R20,R34
0BC5 14 A3              STM R20,R24             ! R24m=target address
0BC7 A9 80 81           LDM R20,=INPBUF         ! R20m=source address
0BCA 52 91              TSM R22                 ! handle zero-length string
0BCC F7 04              JZR EXPE20
0BCE 1E C6 FB           JSB X36,L161$           ! use CMOVE to move string
0BD1 02
0BD2          !
0BD2 52 14 E5 EXPE20    PUMD R22,+R24           ! store 2 nulls at the end
0BD5 9E                 RTN                     !  of the string
0BD6 85       L127      BYT 205
0BD7 51 55 45           ASP "QUERY"
0BDA 52 D9
0BDC 9C 0B              DEF L126
0BDE B8 05    QUERY     DEF DOCOL
0BE0 9A 06              DEF TIB
0BE2 5D 05              DEF AT
0BE4 83 00              DEF LIT
0BE6 60 00              BYT 140,0               ! 96 CHARACTERS INPUT
0BE8 A5 0B              DEF EXPEC
0BEA 23 06              DEF ZERO
0BEC 63 07              DEF IN
0BEE 81 05              DEF STORE
0BF0 12 04              DEF SEMIS
0BF2          !
```

```
                !       "NULL" FILL                       35
0BF2            ! THE NULL OPERATION(ASCII 0) STOPS INTERPRETATION/COMPILATIO
0BF2            ! AT END OF A TERMINAL INPUT LINE, OR A DISK SCREEN. ALL DISK
0BF2            ! BUFFERS MUST TERMINATE WITH NULLS, AND 'EXPECT' PLACES NULL
0BF2            ! AFTER EACH TERMINAL INPUT LINE.
0BF2 C1         L300    BYT 301
0BF3 80                 BYT 200                 ! ASCII NULL (X)
0BF4 D6 0B              DEF L127
0BF6 B8 05      NULL    DEF DOCOL
0BF8 5A 07              DEF BLK
0BFA 5D 05              DEF AT
0BFC DE 00              DEF ZBRAN
0BFE 26 00              BYT 46,0                ! XXJ2-
0C00 2B 06              DEF ONE
0C02 5A 07              DEF BLK
0C04 31 05              DEF PSTOR
0C06 23 06              DEF ZERO
0C08 63 07              DEF IN
0C0A 81 05              DEF STORE
0C0C 5A 07              DEF BLK
0C0E 5D 05              DEF AT
0C10 7E 06              DEF BSCR
0C12 1E 12              DEF MOD
0C14 59 04              DEF ZEQU
0C16 DE 00              DEF ZBRAN
0C18 08 00              BYT 10,0                ! XXJ1-
0C1A A7 09              DEF QEXEC
0C1C 40 04              DEF FROMR
0C1E 03 05              DEF DROP
0C20 B1 00      XXJ1    DEF BRAN
0C22 06 00              BYT 6,0
0C24 40 04      XXJ2    DEF FROMR
0C26 03 05              DEF DROP
0C28 12 04      XXJ4    DEF SEMIS
0C2A 84         L301    BYT 204
0C2B 46 49 4C           ASP "FILL"              ! FILL
0C2E CC
0C2F F2 0B              DEF L300
0C31 B8 05      FILL    DEF DOCOL
0C33 10 05              DEF SWAP
0C35 33 04              DEF TOR
0C37 F3 04              DEF OVER
0C39 93 05              DEF CSTOR
0C3B 25 05              DEF DUP
0C3D F1 07              DEF ONEP
0C3F 40 04              DEF FROMR
0C41 2B 06              DEF ONE
0C43 4B 08              DEF SUB
0C45 E8 02              DEF CMOVE
0C47 12 04              DEF SEMIS
0C49            !
```

```
                                                          36
                !       ERASE BLANKS HOLD PAD
0C49 85         L302    BYT 205
0C4A 45 52 41           ASP "ERASE"             ! ERASE
0C4D 53 C5
0C4F 2A 0C              DEF L301
0C51 B8 05      ERASE   DEF DOCOL
0C53 23 06              DEF ZERO
0C55 31 0C              DEF FILL
0C57 12 04              DEF SEMIS
0C59 86         L303    BYT 206
0C5A 42 4C 41           ASP "BLANKS"            ! BLANKS
0C5D 4E 4B D3
0C60 49 0C              DEF L302
0C62 B8 05      BLANK   DEF DOCOL
0C64 44 06              DEF BL
0C66 31 0C              DEF FILL
0C68 12 04              DEF SEMIS
0C6A 84         L304    BYT 204
0C6B 48 4F 4C           ASP "HOLD"              ! HOLD
0C6E C4
0C6F 59 0C              DEF L303
0C71 B8 05      HOLD    DEF DOCOL
0C73 83 00              DEF LIT
0C75 FF FF              BYT 377,377
0C77 E8 07              DEF HLD
0C79 31 05              DEF PSTOR
0C7B E8 07              DEF HLD
0C7D 5D 05              DEF AT
0C7F 93 05              DEF CSTOR
0C81 12 04              DEF SEMIS
0C83 83         L305    BYT 203
0C84 50 41 C4           ASP "PAD"               ! PAD
0C87 6A 0C              DEF L304
0C89 B8 05      PAD     DEF DOCOL
0C8B 0E 08              DEF HERE
0C8D 83 00              DEF LIT
0C8F 44 00              BYT 104,0
0C91 84 04              DEF PLUS
0C93 12 04              DEF SEMIS
0C95            !
```

```
                                                                 37
                  !         WORD
0C95 84           L306      BYT 204
0C96 57 4F 52               ASP "WORD"            ! WORD
0C99 C4
0C9A 83 0C                  DEF L305
0C9C B8 05        WORD      DEF DOCOL
0C9E 5A 07                  DEF BLK
0CA0 5D 05                  DEF AT
0CA2 DE 00                  DEF ZBRAN
0CA4 0C 00                  BYT 14,0              ! XXI1
0CA6 5A 07                  DEF BLK
0CA8 5D 05                  DEF AT
0CAA 64 13                  DEF BLOCK
0CAC B1 00                  DEF BRAN
0CAE 06 00                  BYT 6,0               ! XXI2
0CB0 9A 06        XXI1      DEF TIB
0CB2 5D 05                  DEF AT
0CB4 63 07        XXI2      DEF IN
0CB6 5D 05                  DEF AT
0CB8 84 04                  DEF PLUS
0CBA 10 05                  DEF SWAP
0CBC 25 02                  DEF ENCL
0CBE 0E 08                  DEF HERE
0CC0 83 00                  DEF LIT
0CC2 22 00                  BYT 42,0
0CC4 62 0C                  DEF BLANK
0CC6 63 07                  DEF IN
0CC8 31 05                  DEF PSTOR
0CCA F3 04                  DEF OVER
0CCC 4B 08                  DEF SUB
0CCE 33 04                  DEF TOR
0CD0 4C 04                  DEF R
0CD2 0E 08                  DEF HERE
0CD4 93 05                  DEF CSTOR
0CD6 84 04                  DEF PLUS
0CD8 0E 08                  DEF HERE
0CDA F1 07                  DEF ONEP
0CDC 40 04                  DEF FROMR
0CDE E8 02                  DEF CMOVE
0CE0 12 04                  DEF SEMIS
0CE2              !
```

```
                                                                 38
                  !         (NUMBER)
0CE2 88           L307      BYT 210
0CE3 28 4E 55               ASP "(NUMBER)"        ! (NUMBER)
0CE6 4D 42 45
0CE9 52 A9
0CEB 95 0C                  DEF L306
0CED B8 05        PNUMB     DEF DOCOL
0CEF F1 07                  DEF ONEP
0CF1 25 05                  DEF DUP
0CF3 33 04                  DEF TOR
0CF5 6E 05                  DEF CAT
0CF7 B7 07                  DEF BASE
0CF9 5D 05                  DEF AT
0CFB 72 01                  DEF DIGIT
0CFD DE 00                  DEF ZBRAN
0CFF 2C 00                  BYT 54,0
0D01 10 05                  DEF SWAP
0D03 B7 07                  DEF BASE
0D05 5D 05                  DEF AT
0D07 0A 03                  DEF USTAR
0D09 03 05                  DEF DROP
0D0B A2 08                  DEF ROT
0D0D B7 07                  DEF BASE
0D0F 5D 05                  DEF AT
0D11 0A 03                  DEF USTAR
0D13 91 04                  DEF DPLUS
0D15 C1 07                  DEF DPL
0D17 5D 05                  DEF AT
0D19 F1 07                  DEF ONEP
0D1B DE 00                  DEF ZBRAN
0D1D 08 00                  BYT 10,0              ! XXG5-
0D1F 2B 06                  DEF ONE
0D21 C1 07                  DEF DPL
0D23 31 05                  DEF PSTOR
0D25 40 04        XXG5      DEF FROMR
0D27 B1 00                  DEF BRAN
0D29 C6 FF                  BYT 306,377           ! XXF3
0D2B 40 04        XXG4      DEF FROMR
0D2D 12 04                  DEF SEMIS
0D2F              !
```

```
                                                                 39
                  !         [NUMBER] NUMBER
0D2F 88           L308      BYT 210
0D30 5B 4E 55               ASP "[NUMBER]"
0D33 4D 42 45
0D36 52 DD
0D38 E2 0C                  DEF L307
0D3A B8 05        BNUMB     DEF DOCOL
0D3C 23 06                  DEF ZERO
0D3E 23 06                  DEF ZERO
0D40 A2 08                  DEF ROT
0D42 25 05                  DEF DUP
0D44 F1 07                  DEF ONEP
0D46 6E 05                  DEF CAT
0D48 83 00                  DEF LIT
0D4A 2D 00                  BYT 55,0
0D4C 5B 08                  DEF EQUAL
0D4E 25 05                  DEF DUP
0D50 33 04                  DEF TOR
0D52 84 04                  DEF PLUS
0D54 83 00                  DEF LIT
0D56 FF FF                  BYT 377,377
0D58 C1 07        XXF6      DEF DPL
0D5A 81 05                  DEF STORE
0D5C ED 0C                  DEF PNUMB
0D5E 25 05                  DEF DUP
0D60 6E 05                  DEF CAT
0D62 44 06                  DEF BL
0D64 4B 08                  DEF SUB
0D66 DE 00                  DEF ZBRAN
0D68 16 00                  BYT 26,0              ! XXF7-
0D6A 25 05                  DEF DUP
0D6C 6E 05                  DEF CAT
0D6E 83 00                  DEF LIT
0D70 2E 00                  BYT 56,0
0D72 4B 08                  DEF SUB
0D74 23 06                  DEF ZERO
0D76 75 09                  DEF QERR
0D78 23 06                  DEF ZERO
0D7A B1 00                  DEF BRAN
0D7C DC FF                  BYT 334,377           ! XXF6-
0D7E 03 05        XXF7      DEF DROP
0D80 40 04                  DEF FROMR
0D82 DE 00                  DEF ZBRAN
0D84 04 00                  BYT 4,0
0D86 D5 04                  DEF DMINU
0D88 12 04        XXFA      DEF SEMIS
0D8A 86           L308.5    BYT 206
0D8B 4E 55 4D               ASP "NUMBER"
0D8E 42 45 D2
0D91 2F 0D                  DEF L308
0D93 B8 05        NUMB      DEF DOCOL
0D95 2D 07                  DEF SNUMB
0D97 5D 05                  DEF AT
0D99 95 00                  DEF EXEC
0D9B 12 04                  DEF SEMIS
0D9D              !
```

```
                                                                 40
                  !         -FIND
0D9D 85           L309      BYT 205
0D9E 2D 46 49               ASP "-FIND"           ! -FIND
0DA1 4E C4
0DA3 8A 0D                  DEF L308.5
0DA5 B8 05        DFIND     DEF DOCOL
0DA7 44 06                  DEF BL
0DA9 9C 0C                  DEF WORD
0DAB 0E 08                  DEF HERE              ! SEARCH CONTEXT VOCABULARY
0DAD 92 07                  DEF CONT
0DAF 5D 05                  DEF AT
0DB1 5D 05                  DEF AT
0DB3 B2 01                  DEF PFIND
0DB5 25 05                  DEF DUP
0DB7 59 04                  DEF ZEQU
0DB9 DE 00                  DEF ZBRAN
0DBB 0E 00                  BYT 16,0
0DBD 03 05                  DEF DROP
0DBF 0E 08                  DEF HERE
0DC1 83 00                  DEF LIT               ! SEARCH FORTH VOCABULARY
0DC3 EE 1B                  DEF FPTR
0DC5 5D 05                  DEF AT
0DC7 B2 01                  DEF PFIND
0DC9 12 04        XXE3      DEF SEMIS
0DCB 87           L311      BYT 207
0DCC 28 41 42               ASP "(ABORT)"         ! (ABORT)
0DCF 4F 52 54
0DD2 A9
0DD3 9D 0D                  DEF L309
0DD5 B8 05        PABOR     DEF DOCOL
0DD7 3A 07                  DEF SABOR
0DD9 5D 05                  DEF AT
0DDB 95 00                  DEF EXEC
0DDD 12 04                  DEF SEMIS
0DDF              !
```

```
                   !          ERROR ID.                                41
0DDF 85           L312      BYT 205
0DE0 45 52 52               ASP "ERROR"            ! ERROR
0DE3 4F D2
0DE5 CB 0D                  DEF L311
0DE7 B8 05        ERROR     DEF DOCOL
0DE9 B4 06                  DEF WARN
0DEB 5D 05                  DEF AT
0DED 6F 04                  DEF ZLESS
0DEF DE 00                  DEF ZBRAN
0DF1 04 00                  BYT 4,0                ! XXN4
0DF3 D5 0D                  DEF PABOR
0DF5 0E 08        XXN4      DEF HERE
0DF7 E5 0A                  DEF COUNT
0DF9 F8 0A                  DEF TYPE
0DFB 59 0B                  DEF PDOTQ
0DFD 03                     BYT 3
0DFE 20 3F 20               ASC " ? "
0E01 17 15                  DEF MESS
0E03 EF 03                  DEF SPSTO
0E05 63 07                  DEF IN
0E07 5D 05                  DEF AT
0E09 5A 07                  DEF BLK
0E0B 5D 05                  DEF AT
0E0D 39 10                  DEF QUIT
0E0F 12 04                  DEF SEMIS
0E11 83           L313      BYT 203
0E12 49 44 AE               ASP "ID."
0E15 DF 0D                  DEF L312
0E17 B8 05        IDDOT     DEF DOCOL
0E19 89 0C                  DEF PAD
0E1B 83 00                  DEF LIT
0E1D 20 00                  BYT 40,0
0E1F 83 00                  DEF LIT
0E21 5F 00                  BYT 137,0
0E23 31 0C                  DEF FILL
0E25 25 05                  DEF DUP
0E27 4D 09                  DEF PFA
0E29 19 09                  DEF LFA
0E2B F3 04                  DEF OVER
0E2D 4B 08                  DEF SUB
0E2F 89 0C                  DEF PAD
0E31 10 05                  DEF SWAP
0E33 E8 02                  DEF CMOVE
0E35 89 0C                  DEF PAD
0E37 E5 0A                  DEF COUNT
0E39 83 00                  DEF LIT
0E3B 1F 00                  BYT 37,0
0E3D 69 03                  DEF AND
0E3F F8 0A                  DEF TYPE
0E41 BF 08                  DEF SPACE
0E43 12 04                  DEF SEMIS
0E45              !
```

```
                   !          CREATE                                   42
0E45 86           L314      BYT 206
0E46 43 52 45               ASP "CREATE"
0E49 41 54 C5
0E4C 11 0E                  DEF L313
0E4E B8 05        CREAT     DEF DOCOL
0E50 A5 0D                  DEF DFIND
0E52 DE 00                  DEF ZBRAN
0E54 10 00                  BYT 20,0               ! XXD2
0E56 03 05                  DEF DROP
0E58 37 09                  DEF NFA
0E5A 17 0E                  DEF IDDOT
0E5C 83 00                  DEF LIT
0E5E 04 00                  BYT 4,0
0E60 17 15                  DEF MESS
0E62 BF 08                  DEF SPACE
0E64 0E 08        XXD2      DEF HERE
0E66 25 05                  DEF DUP
0E68 6E 05                  DEF CAT
0E6A A6 06                  DEF WIDTH
0E6C 5D 05                  DEF AT
0E6E 25 11                  DEF MIN
0E70 F1 07                  DEF ONEP
0E72 1E 08                  DEF ALLOT
0E74 25 05                  DEF DUP
0E76 83 00                  DEF LIT
0E78 A0 00                  BYT 240,0
0E7A 48 05                  DEF TOGGL
0E7C 0E 08                  DEF HERE
0E7E 2B 06                  DEF ONE
0E80 4B 08                  DEF SUB
0E82 83 00                  DEF LIT
0E84 80 00                  BYT 200,0
0E86 48 05                  DEF TOGGL
0E88 09 09                  DEF LATES
0E8A 2A 08                  DEF COMMA
0E8C A0 07                  DEF CURR
0E8E 5D 05                  DEF AT
0E90 81 05                  DEF STORE
0E92 0E 08                  DEF HERE
0E94 FE 07                  DEF TWOP
0E96 2A 08                  DEF COMMA
0E98 12 04                  DEF SEMIS
0E9A              !
```

```
                   !          [COMPILE] LITERAL DLITERAL               43
0E9A C9           L315      BYT 311
0E9B 5B 43 4F               ASP "[COMPILE]"        ! [COMPILE]
0E9E 4D 50 49
0EA1 4C 45 DD
0EA4 45 0E                  DEF L314
0EA6 B8 05        BCOMP     DEF DOCOL
0EA8 A5 0D                  DEF DFIND
0EAA 59 04                  DEF ZEQU
0EAC 23 06                  DEF ZERO
0EAE 75 09                  DEF QERR
0EB0 03 05                  DEF DROP
0EB2 29 09                  DEF CFA
0EB4 2A 08                  DEF COMMA
0EB6 12 04                  DEF SEMIS
0EB8 C7           L316      BYT 307
0EB9 4C 49 54               ASP "LITERAL"          ! LITERAL
0EBC 45 52 41
0EBF CC
0EC0 9A 0E                  DEF L315
0EC2 B8 05        LITER     DEF DOCOL
0EC4 AC 07                  DEF STATE
0EC6 5D 05                  DEF AT
0EC8 DE 00                  DEF ZBRAN
0ECA 08 00                  BYT 10,0
0ECC 08 0A                  DEF COMP
0ECE 83 00                  DEF LIT
0ED0 2A 08                  DEF COMMA
0ED2 12 04        XXD6      DEF SEMIS
0ED4 C8           L317      BYT 310
0ED5 44 4C 49               ASP "DLITERAL"         ! DLITERAL
0ED8 54 45 52
0EDB 41 CC
0EDD B8 0E                  DEF L316
0EDF B8 05        DLITE     DEF DOCOL
0EE1 AC 07                  DEF STATE
0EE3 5D 05                  DEF AT
0EE5 DE 00                  DEF ZBRAN
0EE7 08 00                  BYT 10,0
0EE9 10 05                  DEF SWAP
0EEB C2 0E                  DEF LITER
0EED C2 0E                  DEF LITER
0EEF 12 04        XXN5      DEF SEMIS
0EF1              !
```

```
                   !          U< ?STACK                                44
0EF1 82           L318      BYT 202
0EF2 55 BC                  ASP "U<"
0EF4 D4 0E                  DEF L317
0EF6 B8 05        ULESS     DEF DOCOL              ! UNSIGNED LESS THAN NEEDED
0EF8 33 04                  DEF TOR                ! FOR '?STACK'
0EFA 23 06                  DEF ZERO
0EFC 40 04                  DEF FROMR
0EFE 23 06                  DEF ZERO
0F00 D5 04                  DEF DMINU
0F02 91 04                  DEF DPLUS
0F04 10 05                  DEF SWAP
0F06 03 05                  DEF DROP
0F08 6F 04                  DEF ZLESS
0F0A 12 04                  DEF SEMIS
0F0C 86           L319      BYT 206
0F0D 3F 53 54               ASP "?STACK"           ! ?STACK
0F10 41 43 CB
0F13 F1 0E                  DEF L318
0F15 B8 05        QSTAC     DEF DOCOL              ! ERROR CHECK
0F17 83 00                  DEF LIT
0F19 34 28                  DEF XSOM2
0F1B DF 03                  DEF SPAT
0F1D F6 0E                  DEF ULESS
0F1F 2B 06                  DEF ONE
0F21 75 09                  DEF QERR
0F23 DF 03                  DEF SPAT
0F25 0E 08                  DEF HERE
0F27 83 00                  DEF LIT
0F29 80 00                  BYT 200,0
0F2B 84 04                  DEF PLUS
0F2D F6 0E                  DEF ULESS
0F2F 33 06                  DEF TWO
0F31 75 09                  DEF QERR
0F33 12 04                  DEF SEMIS
0F35              !
```

```
                  !        INTERPRET IMMEDIATE                    45
0F35 89           L320     BYT 211
0F36 49 4E 54              ASP "INTERPRET"         ! INTERPRET
0F39 45 52 50
0F3C 52 45 D4
0F3F 0C 0F                 DEF L319
0F41 B8 05        INTER    DEF DOCOL
0F43 A5 0D                 DEF DFIND
0F45 DE 00                 DEF ZBRAN
0F47 1E 00                 BYT 36,0
0F49 AC 07                 DEF STATE
0F4B 5D 05                 DEF AT
0F4D 6F 08                 DEF LESS
0F4F DE 00                 DEF ZBRAN
0F51 0A 00                 BYT 12,0
0F53 29 09                 DEF CFA
0F55 2A 08                 DEF COMMA
0F57 B1 00                 DEF BRAN
0F59 06 00                 BYT 6,0
0F5B 29 09        XXE5     DEF CFA
0F5D 95 00                 DEF EXEC
0F5F 15 0F        XXE6     DEF QSTAC
0F61 B1 00                 DEF BRAN
0F63 1C 00                 BYT 34,0
0F65 0E 08        XXEA     DEF HERE
0F67 93 0D                 DEF NUMB
0F69 C1 07                 DEF DPL
0F6B 5D 05                 DEF AT
0F6D F1 07                 DEF ONEP
0F6F DE 00                 DEF ZBRAN
0F71 08 00                 BYT 10,0                ! XXF4
0F73 DF 0E                 DEF DLITE
0F75 B1 00                 DEF BRAN
0F77 06 00                 BYT 6,0                 ! XXF5
0F79 03 05        XXF4     DEF DROP
0F7B C2 0E                 DEF LITER
0F7D 15 0F        XXF5     DEF QSTAC
0F7F B1 00        XXE7     DEF BRAN
0F81 C2 FF                 BYT 302,377
0F83 12 04                 DEF SEMIS
0F85 89           L321     BYT 211
0F86 49 4D 4D              ASP "IMMEDIATE"         ! IMMEDIATE
0F89 45 44 49
0F8C 41 54 C5
0F8F 35 0F                 DEF L320
0F91 B8 05        IMMED    DEF DOCOL
0F93 09 09                 DEF LATES
0F95 83 00                 DEF LIT
0F97 40 00                 BYT 100,0
0F99 48 05                 DEF TOGGL
0F9B 12 04                 DEF SEMIS
0F9D              !
```

```
                  !         VOCABULARY                            46
0F9D 8A           L322     BYT 212
0F9E 56 4F 43              ASP "VOCABULARY"        ! VOCABULARY
0FA1 41 42 55
0FA4 4C 41 52
0FA7 D9
0FA8 85 0F                 DEF L321
0FAA B8 05        VOCAB    DEF DOCOL
0FAC AB 0A                 DEF BUILD
0FAE 83 00                 DEF LIT
0FB0 81 A0                 BYT 201,240
0FB2 2A 08                 DEF COMMA
0FB4 A0 07                 DEF CURR
0FB6 5D 05                 DEF AT
0FB8 25 05                 DEF DUP                 ! CORRECT FOR FORTH
0FBA 83 00                 DEF LIT                 ! VOCAB POINTER IN RAM
0FBC EE 1B                 DEF FPTR
0FBE 5B 08                 DEF EQUAL
0FC0 DE 00                 DEF ZBRAN
0FC2 08 00                 BYT 10,0.
0FC4 03 05                 DEF DROP
0FC6 83 00                 DEF LIT
0FC8 06 10                 DEF XXF+6
0FCA 33 06                 DEF TWO                 ! BRANCH TO HERE
0FCC 4B 08                 DEF SUB
0FCE 2A 08                 DEF COMMA
0FD0 0E 08                 DEF HERE
0FD2 D8 06                 DEF VOCL
0FD4 5D 05                 DEF AT
0FD6 2A 08                 DEF COMMA
0FD8 D8 08                 DEF VOCL
0FDA 81 05                 DEF STORE
0FDC BB 0A                 DEF DOES
0FDE FE 07        DOVOC    DEF TWOP
0FE0 25 05                 DEF DUP                 ! IF FORTH VOCABULARY,
0FE2 83 00                 DEF LIT
0FE4 06 10                 DEF XXF+6               ! USE FPTR
0FE6 5B 08                 DEF EQUAL               ! OTHERWISE DON'T
0FE8 DE 00                 DEF ZBRAN
0FEA 08 00                 BYT 10,0
0FEC 03 05                 DEF DROP
0FEE 83 00                 DEF LIT
0FF0 EE 1B                 DEF FPTR
0FF2 92 07                 DEF CONT
0FF4 81 05                 DEF STORE
0FF6 12 04                 DEF SEMIS
0FF8              !
```

```
                  !        FORTH DEFINITIONS ( QUIT               47
0FF8 C5           L323     BYT 305
0FF9 46 4F 52              ASP "FORTH"             ! FORTH
0FFC 54 C8
0FFE 9D 0F                 DEF L322
1000 C7 0A        FORTH    DEF DODOE
1002 DE 0F                 DEF DOVOC
1004 81 A0                 BYT 201,240
1006 00 00        XXF+6    BYT 0,0                 ! TERMINATE FORTH LINKAGE HERE
1008 00 00        XXVOC    BYT 0,0                 ! VOCABULARY LINK
100A 8B           L324     BYT 213
100B 44 45 46              ASP "DEFINITIONS"       ! DEFINITIONS
100E 49 4E 49
1011 54 49 4F
1014 4E D3
1016 F8 0F                 DEF L323
1018 B8 05        DEFIN    DEF DOCOL
101A 92 07                 DEF CONT
101C 5D 05                 DEF AT
101E A0 07                 DEF CURR
1020 81 05                 DEF STORE
1022 12 04                 DEF SEMIS
1024 C1           L325     BYT 301
1025 A8                    ASP "("                 ! (
1026 0A 10                 DEF L324
1028 B8 05        PAREN    DEF DOCOL
102A 83 00                 DEF LIT
102C 29 00                 BYT 51,0
102E 9C 0C                 DEF WORD
1030 12 04                 DEF SEMIS
1032 84           L326     BYT 204
1033 51 55 49              ASP "QUIT"              ! QUIT
1036 D4
1037 24 10                 DEF L325
1039 B8 05        QUIT     DEF DOCOL
103B 23 06                 DEF ZERO
103D 5A 07                 DEF BLK
103F 81 05                 DEF STORE
1041 1E 0A                 DEF LBRAC
1043 01 04        XXB1     DEF RPSTO
1045 DE 0B                 DEF QUERY
1047 41 0F                 DEF INTER
1049 AC 07                 DEF STATE
104B 5D 05                 DEF AT
104D 59 04                 DEF ZEQU
104F 1F 07                 DEF OKFLG
1051 5D 05                 DEF AT
1053 69 03                 DEF AND
1055 DE 00                 DEF ZBRAN
1057 08 00                 BYT 10,0                ! XXB2
1059 59 0B                 DEF PDOTQ
105B 03                    BYT 3
105C 4F 4B 20              ASC "OK "
105F B1 00        XXB2     DEF BRAN
1061 E2 FF                 BYT 342,377
1063 12 04                 DEF SEMIS
1065              !
```

```
                  !         ABORT                                 48
1065 85           L327     BYT 205
1066 41 42 4F              ASP "ABORT"             ! ABORT
1069 52 D4
106B 32 10                 DEF L326
106D B8 05        ABORT    DEF DOCOL
106F EF 03        AB+2     DEF SPSTO
1071 83 00                 DEF LIT
1073 36 28                 DEF XSO
1075 87 06                 DEF SZERO
1077 81 05                 DEF STORE
1079 C1 12                 DEF MTBUF
107B 69 0A                 DEF DEC
107D BF 08                 DEF SPACE
107F D6 02                 DEF CR
1081 59 0B                 DEF PDOTQ
1083 0F                    BYT 17
1084 48 50 37              ASC "HP75 FORTH 1.0 "
1087 35 20 46
108A 4F 52 54
108D 48 20 31
1090 2E 30 20
1093 00 10                 DEF FORTH
1095 18 10                 DEF DEFIN
1097 39 10                 DEF QUIT
1099 12 04                 DEF SEMIS
109B              !
```

```
                                                                 49
                  !          Startup logic
1098 A1                      BYT 241                  ! CODE ATTRIBUTE BYTE
109C 98           FORTH.     BIN                      ! MUST BE IN BINARY MODE
109D CE C4 4F                JSB =SAVENV              ! R10 relativize & save
10A0 5C B1 A3                LDMD R34,=ROMPTR
10A3 82
10A4 56 A9 0A                LDM R26,=12,0            ! move 12 bytes
10A7 00
10A8 5A A9 73                LDM R32,=OR+22           ! MOV #ORIGIN+22,R3 START MOVI
10AB 00
10AC 1C C3                   ADM R32,R34
10AE 54 B5 71                LDMD R24,X34,OR+20       ! MOV ORIGIN+20,R4 MOVE TO USE
10B1 00
10B2 C3                      ADM R24,R34
10B3 CB 06 00                ADM R24,=6,0             ! ADD #6,R4
10B6 56 1A C3                ADM R26,R32              ! ADD R3,R5 COMPUTE LOOP STOP
10B9 50 1A E1 L4001$         POMD R20,+R32            ! MOV (R3)+,(R4)+
10BC 14 E5                   PUMD R20,+R24
10BE 5A 16 C1                CMM R32,R26
10C1 F4 F6                   JNG L4001$               ! BLT 1$
10C3 40 1C B5                LDMD R0,X34,OR+24        ! MOV ORIGIN+24,RP
10C6 75 00
10C8 C3                      ADM R0,R34
10C9 48 A9 6F                LDM R10,=AB+2            ! MOV #ABORT+2,IP
10CC 10
10CD C3                      ADM R10,R34
10CE 5E A1                   LDM R36,R34              ! INIT FORTH RELOC REG
10D0 4C 08 E1 NEXT           POMD R14,+R10            ! WA<-C(I); I<-I+2
10D3 1C C3                   ADM R14,R34              ! MAKE WA ABSOLUTE
10D5 50 0C E1                POMD R20,+R14            ! CA<-C(WA); WA<-WA+2
10D8 1C C3                   ADM R20,R34              ! MAKE CA ABSOLUTE
10DA 10 C6 00                JSB X20,0                ! EXECUTE PROLOGUE OF THIS WOR
10DD 00
10DE F0 F0                   JMP NEXT
10E0 84           L401       BYT 204
10E1 53 2D 3E                ASP "S->D"               ! SINGLE TO DOUBLE
10E4 C4
10E5 65 10                   DEF L327
10E7 E9 10        STOD       DEF STOD+
10E9 50 93        STOD+      CLM R20
10EB 52 1A A5                LDMD R22,R32
10EE F5 02                   JPS L4011$
10F0 50 8B                   DCM R20
10F2 50 1A E7 L4011$         PUMD R20,-R32
10F5 9E                      RTN
10F6              !
```

```
                                                                       50
10F6              ! NOTE: THIS SYSTEM DOES NOT NEED THE OPERATIONS '+-' AND 'D+
10F6              ! BECAUSE 'M*' AND 'M/' ARE DEFINED IN CODE
10F6 83           L402       BYT 203
10F7 41 42 D3                ASP "ABS"
10FA E0 10                   DEF L401
10FC B8 05        ABS        DEF DOCOL
10FE 25 05                   DEF DUP
1100 6F 04                   DEF ZLESS
1102 DE 00                   DEF ZBRAN
1104 04 00                   BYT 4,0                  ! XXR5
1106 C4 04                   DEF MINUS
1108 12 04        XXR5       DEF SEMIS
110A 84           L403       BYT 204
110B 44 41 42                ASP "DABS"
110E D3
110F F6 10                   DEF L402
1111 B8 05        DABS       DEF DOCOL
1113 25 05                   DEF DUP
1115 6F 04                   DEF ZLESS
1117 DE 00                   DEF ZBRAN
1119 04 00                   BYT 4,0                  ! XXRB
111B D5 04                   DEF DMINU
111D 12 04        XXRB       DEF SEMIS
111F 83           L404       BYT 203
1120 4D 49 CE                ASP "MIN"
1123 0A 11                   DEF L403
1125 B8 05        MIN        DEF DOCOL
1127 F3 04                   DEF OVER
1129 F3 04                   DEF OVER
112B 87 08                   DEF GREAT
112D DE 00                   DEF ZBRAN
112F 04 00                   BYT 4,0                  ! XXR7
1131 10 05                   DEF SWAP
1133 03 05        XXR7       DEF DROP
1135 12 04                   DEF SEMIS
1137 83           L405       BYT 203
1138 4D 41 D8                ASP "MAX"
113B 1F 11                   DEF L404
113D B8 05        MAX        DEF DOCOL
113F F3 04                   DEF OVER
1141 F3 04                   DEF OVER
1143 6F 08                   DEF LESS
1145 DE 00                   DEF ZBRAN
1147 04 00                   BYT 4,0                  ! XXR6
1149 10 05                   DEF SWAP
114B 03 05        XXR6       DEF DROP
114D 12 04                   DEF SEMIS
114F              !
```

```
                                                                 51
                  !          M*
114F 82           L406       BYT 202
1150 4D AA                   ASP "M*"
1152 37 11                   DEF L405
1154 56 11        MSTAR      DEF MSTAR+
1156 50 1A B5 MSTAR+         LDMD R20,X32,TWO.        ! SAVE SIGN
1159 02 00
115B 00 E7                   PUMD R20,-R0
115D F5 05                   JPS L4061$
115F 8D                      TCM R20                  ! GET ABS VALUE
1160 1A B7 02                STMD R20,X32,TWO.
1163 00
1164 52 1A A5 L4061$         LDMD R22,R32
1167 F5 09                   JPS L4062$
1169 50 00 A5                LDMD R20,R0              ! ADJUST SAVED SIGN
116C 8D                      TCM R20
116D A7                      STMD R20,R0
116E 52 8D                   TCM R22                  ! GET ABS VALUE
1170 1A A7                   STMD R22,R32
1172 1E C6 0C L4062$         JSB X36,UMULT            ! MULTIPLY
1175 03
1176 50 00 E1                POMD R20,+R0
1179 F5 14                   JPS L4063$
117B 1A A5                   LDMD R20,R32             ! NEGATE DOUBLE-INTEGER # ON S
117D 8F                      NCM R20
117E 52 B5 02                LDMD R22,X32,TWO.
1181 00
1182 8F                      NCM R22
1183 89                      ICM R22
1184 FA 02                   JNC L4064$
1186 50 89                   ICM R20
1188 50 1A A7 L4064$         STMD R20,R32
118B 52 B7 02                STMD R22,X32,TWO.
118E 00
118F 9E           L4063$     RTN
1190              !
```

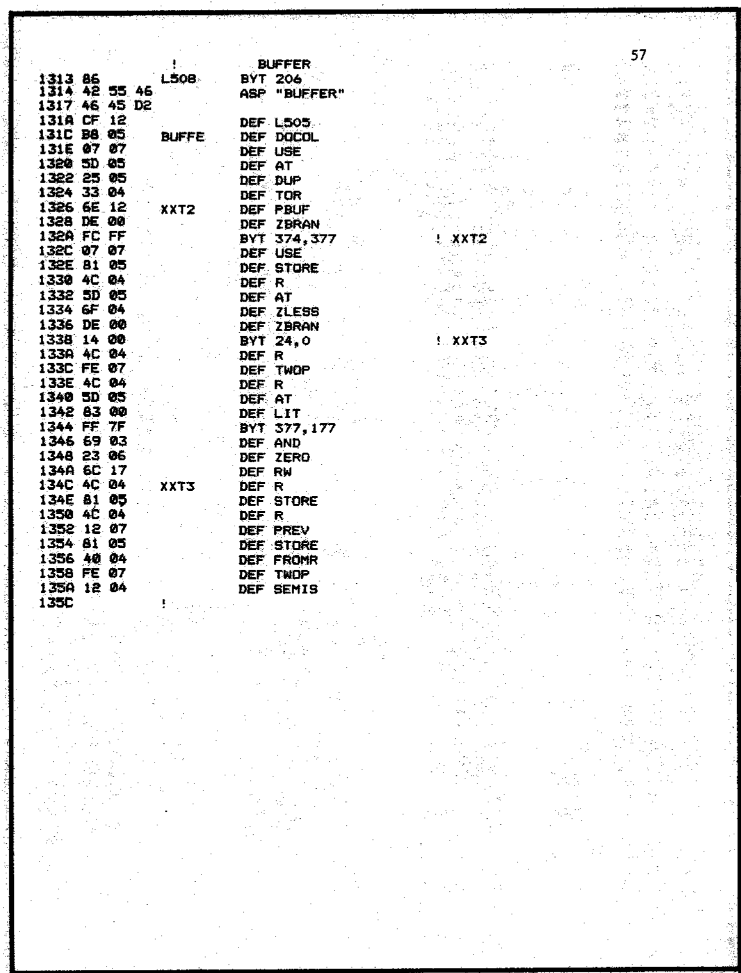
```
                                                                 52
                  !          M/
1190 82           L407       BYT 202
1191 4D AF                   ASP "M/"
1193 4F 11                   DEF L406
1195 97 11        MSLAS      DEF MSLAS+
1197 50 1A B5 MSLAS+         LDMD R20,X32,TWO.        ! SAVE SIGN
119A 02 00
119C 00 E7                   PUMD R20,-R0
119E F6 04                   JNZ L4075$               ! NO SIGN CHANGE
11A0 52 A5                   LDMD R22,R0
11A2 89                      ICM R22
11A3 A7                      STMD R22,R0
11A4 52 00 A5 L4075$         LDMD R22,R0              ! DUPLICATE IT
11A7 E7                      PUMD R22,-R0
11A8 F5 18                   JPS L4071$
11AA              ! TAKE ABS VALUE OF DOUBL-INTEGER DIVDEND
11AA 50 1A B5                LDMD R20,X32,TWO.
11AD 02 00
11AF 8F                      NCM R20
11B0 52 B5 04                LDMD R22,X32,FOUR.
11B3 00
11B4 8D                      TCM R22
11B5 FA 02                   JNC L4076$
11B7 50 89                   ICM R20
11B9 50 1A B7 L4076$         STMD R20,X32,TWO.
11BC 02 00
11BE 52 B7 04                STMD R22,X32,FOUR.
11C1 00
11C2 50 1A A5 L4071$         LDMD R20,R32
11C5 F5 09                   JPS L4072$               ! IS DIVISOR NEGATIVE?
11C7 52 00 A5                LDMD R22,R0              ! IF YES NEGATE QUOTIENT SIGN
11CA 8D                      TCM R22
11CB A7                      STMD R22,R0
11CC 50 8D                   TCM R20                  ! AND TAKE ABS OF DIVISOR
11CE 1A A7                   STMD R20,R32
11D0 1E C6 3F L4072$         JSB X36,UDIV
11D3 03
11D4 50 00 E1                POMD R20,+R0             ! NEGATIVE QUOTIENT?
11D7 F5 04                   JPS L4073$               ! NO
11D9 1A A5                   LDMD R20,R32             ! NEGATE QUOTIENT
11DB 8D                      TCM R20
11DC A7                      STMD R20,R32
11DD 50 00 E1 L4073$         POMD R20,+R0             ! NEGATIVE DIVIDEND?
11E0 F5 08                   JPS L4074$               ! NEGATE REMAINDER
11E2 1A B5 02                LDMD R20,X32,TWO.
11E5 00
11E6 8D                      TCM R20
11E7 B7 02 00                STMD R20,X32,TWO.
11EA 9E           L4074$     RTN
11EB              !
```

```
                                                                  53
                !           * /MOD /
11EB 81         L408      BYT 201
11EC AA                   ASP "*"
11ED 90 11                DEF L407
11EF B8 05      STAR      DEF DOCOL
11F1 54 11                DEF MSTAR
11F3 03 05                DEF DROP
11F5 12 04                DEF SEMIS
11F7 84         L409      BYT 204
11F8 2F 4D 4F             ASP "/MOD"
11FB C4
11FC EB 11                DEF L408
11FE B8 05      SLMOD     DEF DOCOL
1200 33 04                DEF TOR
1202 E7 10                DEF STOD
1204 40 04                DEF FROMR
1206 95 11                DEF MSLAS
1208 12 04                DEF SEMIS
120A 81         L410      BYT 201
120B AF                   ASP "/"
120C F7 11                DEF L409
120E B8 05      SLASH     DEF DOCOL
1210 FE 11                DEF SLMOD
1212 10 05                DEF SWAP
1214 03 05                DEF DROP
1216 12 04                DEF SEMIS
1218            !
```

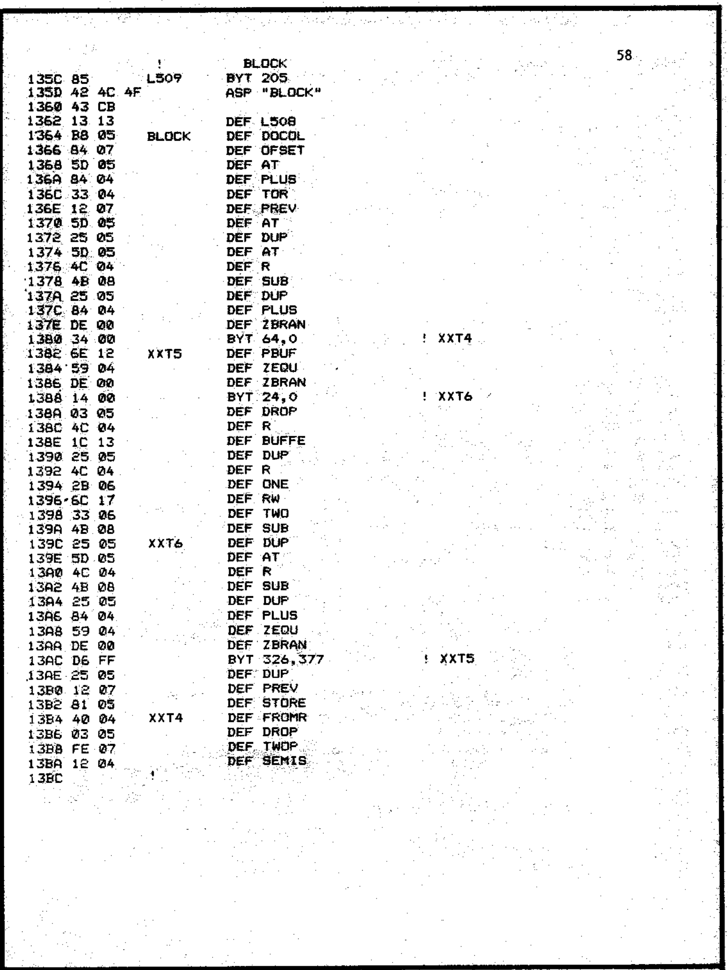
```
                                                                  54
                !            MOD */MOD */ M/MOD
1218 83         L411      BYT 203
1219 4D 4F C4             ASP "MOD"
121C 0A 12                DEF L410
121E B8 05      MOD       DEF DOCOL
1220 FE 11                DEF SLMOD
1222 03 05                DEF DROP
1224 12 04                DEF SEMIS
1226 85         L412      BYT 205
1227 2A 2F 4D             ASP "*/MOD"
122A 4F C4
122C 18 12                DEF L411
122E B8 05      SSMOD     DEF DOCOL
1230 33 04                DEF TOR
1232 54 11                DEF MSTAR
1234 40 04                DEF FROMR
1236 95 11                DEF MSLAS
1238 12 04                DEF SEMIS
123A 82         L413      BYT 202
123B 2A AF                ASP "*/"
123D 26 12                DEF L412
123F B8 05      SSLA      DEF DOCOL
1241 2E 12                DEF SSMOD
1243 10 05                DEF SWAP
1245 03 05                DEF DROP
1247 12 04                DEF SEMIS
1249 85         L414      BYT 205
124A 4D 2F 4D             ASP "M/MOD"
124D 4F C4
124F 3A 12                DEF L413
1251 B8 05      MSMOD     DEF DOCOL
1253 33 04                DEF TOR
1255 23 06                DEF ZERO
1257 4C 04                DEF R
1259 3D 03                DEF USLAS
125B 40 04                DEF FROMR
125D 10 05                DEF SWAP
125F 33 04                DEF TOR
1261 3D 03                DEF USLAS
1263 40 04                DEF FROMR
1265 12 04                DEF SEMIS
1267            !
```

```
                                                                  55
                !           +BUF UPDATE EMPTY-BUFFERS
1267 84         L502      BYT 204
1268 2B 42 55             ASP "+BUF"
126B C6
126C 49 12                DEF L414
126E B8 05      PBUF      DEF DOCOL
1270 72 06                DEF BBUF
1272 83 00                DEF LIT
1274 04 00                BYT 4,0
1276 84 04                DEF PLUS
1278 84 04                DEF PLUS
127A 25 05                DEF DUP
127C 66 06                DEF LIMIT
127E 5B 08                DEF EQUAL
1280 DE 00                DEF ZBRAN
1282 06 00                BYT 6,0                 ! XXT1
1284 03 05                DEF DROP
1286 5A 06                DEF FIRST
1288 25 05      XXT1      DEF DUP
128A 12 07                DEF PREV
128C 5D 05                DEF AT
128E 4B 08                DEF SUB
1290 12 04                DEF SEMIS
1292 86         L503      BYT 206
1293 55 50 44             ASP "UPDATE"
1296 41 54 C5
1299 67 12                DEF L502
129B B8 05      UPDAT     DEF DOCOL
129D 12 07                DEF PREV
129F 5D 05                DEF AT
12A1 5D 05                DEF AT
12A3 83 00                DEF LIT
12A5 00 80                BYT 0,200
12A7 7A 03                DEF OR
12A9 12 07                DEF PREV
12AB 5D 05                DEF AT
12AD 81 05                DEF STORE
12AF 12 04                DEF SEMIS
12B1 8D         L504      BYT 215
12B2 45 4D 50             ASP "EMPTY-BUFFERS"
12B5 54 59 2D
12B8 42 55 46
12BB 46 45 52
12BE D3
12BF 92 12                DEF L503
12C1 B8 05      MTBUF     DEF DOCOL
12C3 5A 06                DEF FIRST
12C5 66 06                DEF LIMIT
12C7 F3 04                DEF OVER
12C9 4B 08                DEF SUB
12CB 51 0C                DEF ERASE
12CD 12 04                DEF SEMIS
12CF            !
```

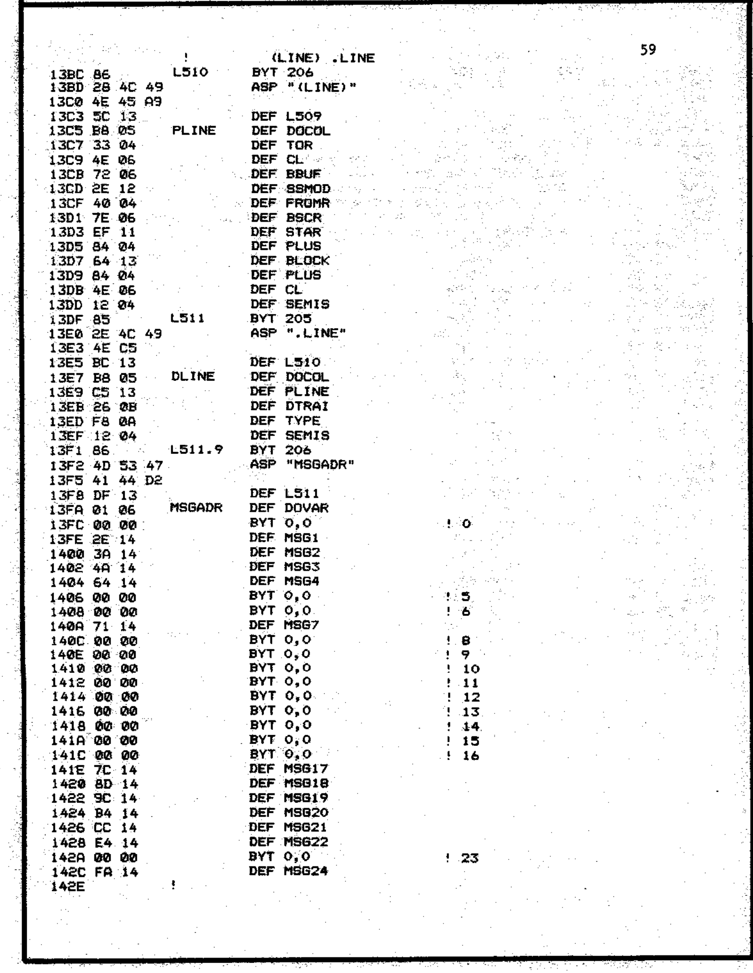
```
                                                                  56
                !            FLUSH
12CF 85         L505      BYT 205
12D0 46 4C 55             ASP "FLUSH"
12D3 53 C8
12D5 B1 12                DEF L504
12D7 B8 05      FLUSH     DEF DOCOL
12D9 66 06                DEF LIMIT
12DB 5A 06                DEF FIRST
12DD 4C 01                DEF XDO
12DF 62 01      XXTA      DEF I
12E1 5D 05                DEF AT
12E3 6F 04                DEF ZLESS
12E5 DE 00                DEF ZBRAN
12E7 1E 00                BYT 36,0                ! XXT7
12E9 62 01                DEF I
12EB FE 07                DEF TWOP
12ED 62 01                DEF I
12EF 5D 05                DEF AT
12F1 83 00                DEF LIT
12F3 FF 7F                BYT 377,177
12F5 69 03                DEF AND
12F7 25 05                DEF DUP
12F9 33 04                DEF TOR
12FB 23 06                DEF ZERO
12FD 6C 17                DEF RW
12FF 40 04                DEF FROMR
1301 62 01                DEF I
1303 81 05                DEF STORE
1305 72 06      XXT7      DEF BBUF
1307 83 00                DEF LIT
1309 04 00                BYT 4,0
130B 84 04                DEF PLUS
130D 12 01                DEF XPLOO
130F D0 FF                BYT 320,377             ! XXTA
1311 12 04                DEF SEMIS
1313            !
```



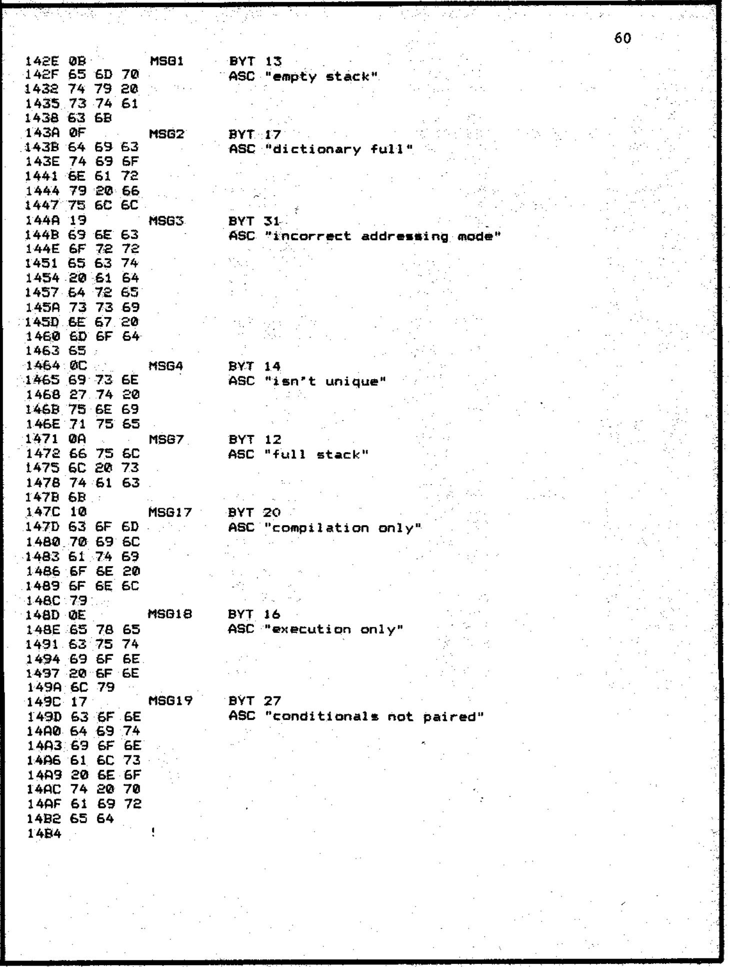
```
                                                          57
                  !          BUFFER
1313 86          L508       BYT 206
1314 42 55 46               ASP "BUFFER"
1317 46 45 D2
131A CF 12                  DEF L505
131C B8 05       BUFFE      DEF DOCOL
131E 07 07                  DEF USE
1320 5D 05                  DEF AT
1322 25 05                  DEF DUP
1324 33 04                  DEF TOR
1326 6E 12       XXT2       DEF PBUF
1328 DE 00                  DEF ZBRAN
132A FC FF                  BYT 374,377           ! XXT2
132C 07 07                  DEF USE
132E 81 05                  DEF STORE
1330 4C 04                  DEF R
1332 5D 05                  DEF AT
1334 6F 04                  DEF ZLESS
1336 DE 00                  DEF ZBRAN
1338 14 00                  BYT 24,0              ! XXT3
133A 4C 04                  DEF R
133C FE 07                  DEF TWOP
133E 4C 04                  DEF R
1340 5D 05                  DEF AT
1342 83 00                  DEF LIT
1344 FF 7F                  BYT 377,177
1346 69 03                  DEF AND
1348 23 06                  DEF ZERO
134A 6C 17                  DEF RW
134C 4C 04       XXT3       DEF R
134E 81 05                  DEF STORE
1350 4C 04                  DEF R
1352 12 07                  DEF PREV
1354 81 05                  DEF STORE
1356 40 04                  DEF FROMR
1358 FE 07                  DEF TWOP
135A 12 04                  DEF SEMIS
135C             !
```



```
                                                          58
                  !          BLOCK
135C 85          L509       BYT 205
135D 42 4C 4F               ASP "BLOCK"
1360 43 CB
1362 13 13                  DEF L508
1364 B8 05       BLOCK      DEF DOCOL
1366 84 07                  DEF OFSET
1368 5D 05                  DEF AT
136A 84 04                  DEF PLUS
136C 33 04                  DEF TOR
136E 12 07                  DEF PREV
1370 5D 05                  DEF AT
1372 25 05                  DEF DUP
1374 5D 05                  DEF AT
1376 4C 04                  DEF R
1378 4B 08                  DEF SUB
137A 25 05                  DEF DUP
137C 84 04                  DEF PLUS
137E DE 00                  DEF ZBRAN
1380 34 00                  BYT 64,0              ! XXT4
1382 6E 12       XXT5       DEF PBUF
1384 59 04                  DEF ZEQU
1386 DE 00                  DEF ZBRAN
1388 14 00                  BYT 24,0              ! XXT6
138A 03 05                  DEF DROP
138C 4C 04                  DEF R
138E 1C 13                  DEF BUFFE
1390 25 05                  DEF DUP
1392 4C 04                  DEF R
1394 2B 06                  DEF ONE
1396 6C 17                  DEF RW
1398 33 06                  DEF TWO
139A 4B 08                  DEF SUB
139C 25 05       XXT6       DEF DUP
139E 5D 05                  DEF AT
13A0 4C 04                  DEF R
13A2 4B 08                  DEF SUB
13A4 25 05                  DEF DUP
13A6 84 04                  DEF PLUS
13A8 59 04                  DEF ZEQU
13AA DE 00                  DEF ZBRAN
13AC D6 FF                  BYT 326,377           ! XXT5
13AE 25 05                  DEF DUP
13B0 12 07                  DEF PREV
13B2 81 05                  DEF STORE
13B4 40 04       XXT4       DEF FROMR
13B6 03 05                  DEF DROP
13B8 FE 07                  DEF TWOP
13BA 12 04                  DEF SEMIS
13BC             !
```



```
                                                          59
                  !          (LINE) .LINE
13BC 86          L510       BYT 206
13BD 28 4C 49               ASP "(LINE)"
13C0 4E 45 A9
13C3 5C 13                  DEF L509
13C5 B8 05       PLINE      DEF DOCOL
13C7 33 04                  DEF TOR
13C9 4E 06                  DEF CL
13CB 72 06                  DEF BBUF
13CD 2E 12                  DEF SSMOD
13CF 40 04                  DEF FROMR
13D1 7E 06                  DEF BSCR
13D3 EF 11                  DEF STAR
13D5 84 04                  DEF PLUS
13D7 64 13                  DEF BLOCK
13D9 84 04                  DEF PLUS
13DB 4E 06                  DEF CL
13DD 12 04                  DEF SEMIS
13DF 85          L511       BYT 205
13E0 2E 4C 49               ASP ".LINE"
13E3 4E C5
13E5 BC 13                  DEF L510
13E7 B8 05       DLINE      DEF DOCOL
13E9 C5 13                  DEF PLINE
13EB 26 0B                  DEF DTRAI
13ED F8 0A                  DEF TYPE
13EF 12 04                  DEF SEMIS
13F1 86          L511.9     BYT 206
13F2 4D 53 47               ASP "MSGADR"
13F5 41 44 D2
13F8 DF 13                  DEF L511
13FA 01 06       MSGADR     DEF DOVAR
13FC 00 00                  BYT 0,0               ! 0
13FE 2E 14                  DEF MSG1
1400 3A 14                  DEF MSG2
1402 4A 14                  DEF MSG3
1404 64 14                  DEF MSG4
1406 00 00                  BYT 0,0               ! 5
1408 00 00                  BYT 0,0               ! 6
140A 71 14                  DEF MSG7
140C 00 00                  BYT 0,0               ! 8
140E 00 00                  BYT 0,0               ! 9
1410 00 00                  BYT 0,0               ! 10
1412 00 00                  BYT 0,0               ! 11
1414 00 00                  BYT 0,0               ! 12
1416 00 00                  BYT 0,0               ! 13
1418 00 00                  BYT 0,0               ! 14
141A 00 00                  BYT 0,0               ! 15
141C 00 00                  BYT 0,0               ! 16
141E 7C 14                  DEF MSG17
1420 8D 14                  DEF MSG18
1422 9C 14                  DEF MSG19
1424 B4 14                  DEF MSG20
1426 CC 14                  DEF MSG21
1428 E4 14                  DEF MSG22
142A 00 00                  BYT 0,0               ! 23
142C FA 14                  DEF MSG24
142E             !
```



```
                                                          60
142E 0B          MSG1       BYT 13
142F 65 6D 70               ASC "empty stack"
1432 74 79 20
1435 73 74 61
1438 63 6B
143A 0F          MSG2       BYT 17
143B 64 69 63               ASC "dictionary full"
143E 74 69 6F
1441 6E 61 72
1444 79 20 66
1447 75 6C 6C
144A 19          MSG3       BYT 31
144B 69 6E 63               ASC "incorrect addressing mode"
144E 6F 72 72
1451 65 63 74
1454 20 61 64
1457 64 72 65
145A 73 73 69
145D 6E 67 20
1460 6D 6F 64
1463 65
1464 0C          MSG4       BYT 14
1465 69 73 6E               ASC "isn't unique"
1468 27 74 20
146B 75 6E 69
146E 71 75 65
1471 0A          MSG7       BYT 12
1472 66 75 6C               ASC "full stack"
1475 6C 20 73
1478 74 61 63
147B 6B
147C 10          MSG17      BYT 20
147D 63 6F 6D               ASC "compilation only"
1480 70 69 6C
1483 61 74 69
1486 6F 6E 20
1489 6F 6E 6C
148C 79
148D 0E          MSG18      BYT 16
148E 65 78 65               ASC "execution only"
1491 63 75 74
1494 69 6F 6E
1497 20 6F 6E
149A 6C 79
149C 17          MSG19      BYT 27
149D 63 6F 6E               ASC "conditionals not paired"
14A0 64 69 74
14A3 69 6F 6E
14A6 61 6C 73
14A9 20 6E 6F
14AC 74 20 70
14AF 61 69 72
14B2 65 64
14B4             !
```

61

```
14B4 17         MSG20     BYT 27
14B5 64 65 66             ASC "definition not finished"
14B8 69 6E 69
14BB 74 69 6F
14BE 6E 20 6E
14C1 6F 74 20
14C4 66 69 6E
14C7 69 73 68
14CA 65 64
14CC 17         MSG21     BYT 27
14CD 69 6E 20             ASC "in protected dictionary"
14D0 70 72 6F
14D3 74 65 63
14D6 74 65 64
14D9 20 64 69
14DC 63 74 69
14DF 6F 6E 61
14E2 72 79
14E4 15         MSG22     BYT 25
14E5 75 73 65             ASC "use only when loading"
14E8 20 6F 6E
14EB 6C 79 20
14EE 77 68 65
14F1 6E 20 6C
14F4 6F 61 64
14F7 69 6E 67
14FA 12         MSG24     BYT 22
14FB 64 65 63             ASC "declare vocabulary"
14FE 6C 61 72
1501 65 20 76
1504 6F 63 61
1507 62 75 6C
150A 61 72 79
150D 87         L512      BYT 207
150E 4D 45 53             ASP "MESSAGE"
1511 53 41 47
1514 C5
1515 F1 13                DEF L511.9
1517            ! : MESSAGE ( msg# - )
1517            ! first decide if it is in the message table
1517            ! after this test, SP will have:
1517            !     either msg# 0       or msg# msg.adr
1517            ! 0 OVER < IF
1517            !   DUP 25 < IF        message adr table has space for 24 entr
1517            !     DUP 2 * MSGADR + @
1517            !   ELSE 0 THEN
1517            ! ELSE 0 THEN
1517            ! now SP has either  msg# 0  or  msg# msg.adr
1517            ! -DUP IF COUNT TYPE BL EMIT DROP
1517            ! ELSE -DUP IF
1517            !     WARNING @ IF
1517            !       4 OFFSET @ B/SCR / - .LINE
1517            !     ELSE
1517            !       ." MSG # " .
1517            !   THEN THEN
1517            ! THEN ;
1517            !
```

62

```
1517 B8 05      MESS      DEF DOCOL
1519 23 06                DEF ZERO
151B F3 04                DEF OVER
151D 6F 08                DEF LESS
151F DE 00                DEF ZBRAN
1521 24 00                BYT 44,0
1523 25 05                DEF DUP
1525 83 00                DEF LIT
1527 19 00                BYT 31,0
1529 6F 08                DEF LESS
152B DE 00                DEF ZBRAN
152D 12 00                BYT 22,0
152F 25 05                DEF DUP
1531 33 06                DEF TWO
1533 EF 11                DEF STAR
1535 FA 13                DEF MSGADR
1537 84 04                DEF PLUS
1539 5D 05                DEF AT
153B B1 00                DEF BRAN
153D 04 00                BYT 4,0
153F 23 06                DEF ZERO
1541 B1 00                DEF BRAN
1543 04 00                BYT 4,0
1545 23 06                DEF ZERO
1547 D0 08                DEF DDUP
1549 DE 00                DEF ZBRAN
154B 10 00                BYT 20,0
154D E5 0A                DEF COUNT
154F F8 0A                DEF TYPE
1551 44 06                DEF BL
1553 8E 02                DEF EMIT
1555 03 05                DEF DROP
1557 B1 00                DEF BRAN
1559 2F 00                BYT 57,0
155B D0 08                DEF DDUP
155D DE 00                DEF ZBRAN
155F 29 00                BYT 51,0
1561 B4 06                DEF WARN
1563 5D 05                DEF AT
1565 DE 00                DEF ZBRAN
1567 16 00                BYT 26,0              ! XXW5
1569 83 00                DEF LIT
156B 04 00                BYT 4,0
156D 84 07                DEF OFSET
156F 5D 05                DEF AT
1571 7E 06                DEF BSCR
1573 0E 12                DEF SLASH
1575 4B 08                DEF SUB
1577 E7 13                DEF DLINE
1579 B1 00      XXW3      DEF BRAN
157B 0D 00                BYT 15,0              ! XXW4
157D 59 0B      XXW5      DEF PDOTQ
157F 06                   BYT 6
1580 4D 53 47             ASC "MSG # "
1583 20 23 20
1586 80 1A                DEF DOT
1588 12 04      XXW4      DEF SEMIS
158A            !
```

63

```
                !         LOAD -->
158A 84         L513      BYT 204
158B 4C 4F 41             ASP "LOAD"
158E C4
158F 0D 15                DEF L512
1591 B8 05      LOAD      DEF DOCOL
1593 5A 07                DEF BLK
1595 5D 05                DEF AT
1597 33 04                DEF TOR
1599 63 07                DEF IN
159B 5D 05                DEF AT
159D 33 04                DEF TOR
159F 23 06                DEF ZERO
15A1 63 07                DEF IN
15A3 81 05                DEF STORE
15A5 7E 06                DEF BSCR
15A7 EF 11                DEF STAR
15A9 5A 07                DEF BLK
15AB 81 05                DEF STORE
15AD 41 0F                DEF INTER
15AF 40 04                DEF FROMR
15B1 63 07                DEF IN
15B3 81 05                DEF STORE
15B5 40 04                DEF FROMR
15B7 5A 07                DEF BLK
15B9 81 05                DEF STORE
15BB 12 04                DEF SEMIS
15BD C3         L514      BYT 303
15BE 2D 2D BE             ASP "-->"
15C1 8A 15                DEF L513
15C3 B8 05      ARROW     DEF DOCOL
15C5 EE 09                DEF QLOAD
15C7 23 06                DEF ZERO
15C9 63 07                DEF IN
15CB 81 05                DEF STORE
15CD 7E 06                DEF BSCR
15CF 5A 07                DEF BLK
15D1 5D 05                DEF AT
15D3 F3 04                DEF OVER
15D5 1E 12                DEF MOD
15D7 4B 08                DEF SUB
15D9 5A 07                DEF BLK
15DB 31 05                DEF PSTOR
15DD 12 04                DEF SEMIS
15DF            !
```

64

```
                !        SCRNAME
15DF 87         L515      BYT 207
15E0 53 43 52             ASP "SCRNAME"
15E3 4E 41 4D
15E6 C5
15E7 BD 15                DEF L514
15E9 B8 05      SCRNAME   DEF DOCOL
15EB            ! : SCRNAME ( blk# - txt.adr count )
15EB            !  BASE @ DECIMAL SWAP  save current base
15EB            !  ABS      take absolute value of blk#
15EB            !  0        make blk# double precision for <#
15EB            !  <# # # # # # "R" HOLD "C" HOLD "S" HOLD #>  make file name
15EB            !  ROT BASE ! ; restore base
15EB B7 07                DEF BASE
15ED 5D 05                DEF AT
15EF 69 0A                DEF DEC
15F1 10 05                DEF SWAP
15F3 FC 10                DEF ABS
15F5 23 06                DEF ZERO
15F7 BD 19                DEF BDIGS
15F9 FB 19                DEF DIG
15FB FB 19                DEF DIG
15FD FB 19                DEF DIG
15FF FB 19                DEF DIG
1601 FB 19                DEF DIG
1603 83 00                DEF LIT
1605 52                   ASC "R"
1606 00                   BYT 0
1607 71 0C                DEF HOLD
1609 83 00                DEF LIT
160B 43                   ASC "C"
160C 00                   BYT 0
160D 71 0C                DEF HOLD
160F 83 00                DEF LIT
1611 53                   ASC "S"
1612 00                   BYT 0
1613 71 0C                DEF HOLD
1615 CC 19                DEF EDIGS
1617 A2 08                DEF ROT
1619 B7 07                DEF BASE
161B 81 05                DEF STORE
161D 12 04                DEF SEMIS
161F            !
```

```
                                                                 65
                  !       RDFILE
161F 86         L515.1   BYT 206
1620 52 44 46            ASP "RDFILE"
1623 49 4C C5
1626 DF 15               DEF L515
1628 2A 16      RDFILE   DEF RDFIL+
162A            RDFIL+   BSZ 0
162A            ! stack has buf.adr, txt.adr, count
162A 50 1A E1            POMD R20,+R32          ! throwaway count
162D E1                  POMD R20,+R32          ! get adr of filename
162E 1C C3               ADM R20,R34
1630 60 10 A5            LDMD R40,R20           ! load file name to R40m
1633 1E C6 B0            JSB X36,SAVFVM
1636 1A
1637 CE 12 22            JSB =FLOPEN
163A F8 6F               JEN BRERR
163C 56 1E A1            LDM R26,R36            ! put file adr in R26
163F 5E 06 E3            POMD R36,-R6
1642 1E C6 C0            JSB X36,GETFVM
1645 1A
1646 50 1A E1            POMD R20,+R32          ! get buffer address to R20m
1649 1C C3               ADM R20,R34
164B 52 10 A1            LDM R22,R20            ! make working copy of buf.adr
164E 55 A8 7F            LDB R25,=127D          ! R25=loop counter
1651 60 A9 20            LDM R40,=40,40,40,40,40,40,40,40
1654 20 20 20
1657 20 20 20
165A 20
165B 60 12 A7 BR10       STMD R40,R22           ! blank out buffer
165E 52 CB 08            ADM R22,=8D,0
1661 00
1662 55 8A               DCB R25
1664 FB F5               JCY BR10
1666 60 A8 0F            LDB R40,=15D           ! R40=loop counter for 16 line
1669 53 93               CLM R23
166B 66 A9 40            LDM R46,=64D,0         ! R46m=constant=max#bytes/line
166E 00
166F            ! R20=buffer address        R26=file address
166F 54 16 A5 BR20       LDMD R24,R26
1672 C9 99 A9            CMM R24,=231,251       ! EOF?
1675 F7 2F               JZR BR90               ! yes
1677 56 89               ICM R26
1679 89                  ICM R26
167A 52 A4               LDBD R22,R26           ! get line length
167C C8 41               CMB R22,=65D
167E F4 02               JNB BR30
1680 26 A1               LDM R22,R46            ! truncate line to 64 bytes
1682 52 18 A3 BR30       STM R22,R30            ! save a copy of line length i
1685 56 89               ICM R26                ! now R26=source address, R22=
1687 55 16 E0 BR40       POBD R25,+R26
168A 10 E4               PUBD R25,+R20
168C 52 8A               DCB R22
168E F6 F7               JNZ BR40
1690            !
```

```
                                                                 66
1690 50 18 C5            SBM R20,R30            ! skip to beginning of next li
1693 26 C3               ADM R20,R46
1695            !
1695 56 18 C5            SBM R26,R30            ! skip to beginning of next li
1698 8B                  DCM R26
1699 6E 16 A4            LDBD R56,R26
169C 6F 92               CLB R57
169E 56 2E C3            ADM R26,R56
16A1 89                  ICM R26
16A2            !
16A2 60 8A               DCB R40
16A4 FB C9               JCY BR20
16A6            !
16A6 50 93      BR90     CLM R20                ! clear error flag
16A8 1A E7               PUMD R20,-R32
16AA 9E                  RTN
16AB            !
16AB 5E 06 E3 BRERR      POMD R36,-R6
16AE 1E C6 C0            JSB X36,GETFVM
16B1 1A
16B2 50 A9 01            LDM R20,=1,0           ! set error flag
16B5 00
16B6 1A A7               STMD R20,R32
16B8 9E                  RTN
16B9            !
16B9 87         L516     BYT 207
16BA 57 52 54            ASP "WRTFILE"
16BD 46 49 4C
16C0 C5
16C1 1F 16               DEF L515.1
16C3 D4 16      WRTFIL   DEF WRTFIL+
16C5            ! WRTFILE ( buf.fvmadr name.fvmadr name.len - errflg )
16C5            ! WRTFIL register usage
16C5            ! R40m=file name
16C5            ! R32m=line length
16C5            ! R34m=line address
16C5            ! R76m=BCD line number, also used as the loop counter for lin
16C5            ! R20m=file type bytes for FREPLN
16C5            ! R32m=# of bytes to delete for DELETE
16C5            ! R22m=local temporary
16C5 5C D5 A3 WRTF40     SBMD R34,=ROMPTR       ! relativize buffer addr
16C8 82
16C9 CE 6D 46            JSB =SYSJSB
16CC 1F 22               DEF FDELLN             ! delete one line
16CE 5C D3 A3            ADMD R34,=ROMPTR       ! make adr absolute again
16D1 82
16D2 F0 44               JMP WRTF60
16D4            !
16D4 1E C6 B0 WRTFIL+    JSB X36,SAVFVM
16D7 1A
16D8 50 1A B5            LDMD R20,X32,TWO.      ! get name address
16DB 02 00
16DD 1C C3               ADM R20,R34
16DF 60 10 A5            LDMD R40,R20           ! move name to R40m
16E2 40 A8 20            LDB R0,=40             ! fill out name with blanks
16E5 1A DA               ADBD R0,R32            ! R0=addr of 1st blank
16E7 41 B1 FB            LDMD R#,=BLANKS
16EA 09
16EB            !
```

```
                                                                 67
16EB 50 B5 04            LDMD R20,X32,FOUR.     ! get buffer address
16EE 00
16EF 1C C3               ADM R20,R34
16F1 A3                  STM R20,R34
16F2 7E 93               CLM R76                ! initialize BCD line#
16F4 50 A9 BE            LDM R20,=276,124       ! load type bytes for text fil
16F7 54
16F8            !
16F8 5A A9 3F WRTF10     LDM R32,=77,0          ! top of line# loop
16FB 00
16FC 55 A8 20            LDB R25,=40            ! for trailing blank test
16FF            !
16FF 52 1A A1 WRTF20     LDM R22,R32            ! top of trailing blank loop
1702 1C C3               ADM R22,R34            ! R22=addr of last char in lin
1704 55 12 D8            CMBD R25,R22           ! trailing blank?
1707 F6 04               JNZ WRTF30             ! nope
1709 5A 8B               DCM R32                ! yes, move back one more
170B F5 F2               JPS WRTF20             ! if any chars left, go around
170D 5A 89      WRTF30   ICM R32                ! adjust to real length
170F F7 B4               JZR WRTF40             ! if empty line, delete it
1711 CE 6D 46            JSB =SYSJSB
1714 59 22               DEF FREPLN             ! insert the line
1716            ! note that FREPLN adjusts R34 for us
1716 F8 49               JEN WRTF80             ! error - not enough memory
1718 5C CB 40 WRTF60     ADM R34,=100,0         ! adjust buffer addr
171B 00
171C 99                  BCD
171D 7E 89               ICM R76                ! inc BCD line number
171F 98                  BIN
1720 C9 16 00            CMM R76,=26,0          ! have we done 16 lines yet?
1723 F6 D3               JNZ WRTF10             ! go do another line
1725            !
1725 CE 5C 1F            JSB =SETPR             ! now delete lines 16-9999
1728 58 A9 16            LDM R30,=26,0          ! line 16
172B 00
172C 5A A9 99            LDM R32,=231,231       ! line 9999
172F 99
1730 CE 6D 46            JSB =SYSJSB
1733 2D 22               DEF DELLNS
1735 52 93               CLM R22                ! clear errflg
1737 5E 06 E3 WRTF70     POMD R36,-R6           ! recover FVM registers
173A 5C E3               POMD R34,-R6
173C 5A E3               POMD R32,-R6
173E 48 E3               POMD R10,-R6
1740 40 E3               POMD R0,-R6
1742 54 E3               POMD R24,-R6           ! get our own rtn adr
1744 50 B1 A3            LDMD R20,=ROMPTR       ! adjust absolute addresses
1747 82
1748 1C C5               SBM R20,R34
174A 40 10 C3            ADM R0,R20
174D 48 C3               ADM R10,R20
174F 5A C3               ADM R32,R20
1751 5C C3               ADM R34,R20
1753 5E C3               ADM R36,R20
1755 54 C3               ADM R24,R20
1757 06 E5               PUMD R24,+R6           ! put rtn adr back
1759 5A CB 04            ADM R32,=4,0           ! clean up SP
175C 00
175D 52 1A A7            STMD R22,R32           ! put errflg onto SP
1760 9E                  RTN
1761            !
```

```
                                                                 68
1761 52 93      WRTF80   CLM R22                ! set errflg
1763 89                  ICM R22
1764 F0 D1               JMP WRTF70
1766            ! WRTFIL could probably be made faster by using FCRNUL
1766            ! and REPLIN or FSREPL instead of FREPLN
1766            !           R/W
1766 83         L517     BYT 203
1767 52 2F D7            ASP "R/W"              ! READ OR WRITE BLOCK, HANDLE
176A B9 16               DEF L516               ! ADDRESS BLOCK# FLAG(1=READ,0
176C B8 05      RW       DEF DOCOL
176E 25 05               DEF DUP
1770 2B 06               DEF ONE
1772 58 06               DEF EQUAL
1774 DE 00               DEF ZBRAN
1776 22 00               BYT 42,0               ! XXS1
1778 03 05               DEF DROP
177A E9 15               DEF SCRNAME
177C 28 16               DEF RDFILE
177E DE 00               DEF ZBRAN
1780 14 00               BYT 24,0               ! XXS2
1782 D6 02               DEF CR
1784 59 0B               DEF PDOTQ
1786 0B                  BYT 13
1787 52 45 41            ASC "READ ERROR "
178A 44 20 45
178D 52 52 4F
1790 52 20
1792 D5 0D               DEF PABOR
1794 B1 00      XXS2     DEF BRAN
1796 23 00               BYT 43,0               ! XXS3
1798 59 04      XXS1     DEF ZEQU
179A DE 00               DEF ZBRAN
179C 1D 00               BYT 35,0               ! XXS4
179E E9 15               DEF SCRNAME
17A0 C3 16               DEF WRTFIL
17A2 DE 00               DEF ZBRAN
17A4 15 00               BYT 25,0               ! XXS5
17A6 D6 02               DEF CR
17A8 59 0B               DEF PDOTQ
17AA 0C                  BYT 14
17AB 57 52 49            ASC "WRITE ERROR "
17AE 54 45 20
17B1 45 52 52
17B4 4F 52 20
17B7 D5 0D               DEF PABOR
17B9            XXS5     BSZ 0
17B9            XXS4     BSZ 0
17B9 12 04      XXS3     DEF SEMIS
17BB            !
```

```
                                                                     69
                   !          PEMIT
17BB               PEMIT    BSZ 0                     ! EMIT CHARACTER
17BB 52 1A E1               POMD R22,+R32             ! POP CHAR OFF COMP. STACK
17BE 1E C6 B0               JSB X36,SAVFVM
17C1 1A
17C2 52 20 A2               STB R22,R40
17C5 CE 14 08               JSB =OUTC40
17C8 5E 06 E3               POMD R36,-R6              ! RESTORE REGISTERS
17CB 1E C6 C0               JSB X36,GETFVM
17CE 1A
17CF 9E                     RTN
17D0               !
17D0 1E C6 B0 PCR           JSB X36,SAVFVM            ! EMIT CR-LF SEQUENCE
17D3 1A
17D4 CE 2C 08               JSB =OUTEOL
17D7 5E 06 E3               POMD R36,-R6
17DA 1E C6 C0               JSB X36,GETFVM
17DD 1A
17DE 9E                     RTN
17DF               !
17DF               PKEY     BSZ 0                     ! GET A KEY
17DF 1E C6 B0               JSB X36,SAVFVM
17E2 1A
17E3 CE 50 07               JSB =GETCHR
17E6 43 92                  CLB R3
17E8 5E 06 E3               POMD R36,-R6
17EB 1E C6 C0               JSB X36,GETFVM
17EE 1A
17EF 42 1A E7               PUMD R2,-R32
17F2 9E                     RTN
17F3               !
17F3               PQTER    BSZ 0                     ! CHECK FOR ANY KEY BUT ATTN
17F3 52 93                  CLM R22                   ! FLAG
17F5 42 B0 84               LDBD R2,=SVCWRD
17F8 82
17F9 F3 0E                  JEV NOKEY
17FB B0 5F 83               LDBD R2,=KEYHIT
17FE C8 80                  CMB R2,=ATTNKY
1800 F7 07                  JZR NOKEY
1802 CE A3 07               JSB =DEQUE
1805 52 A9 01               LDM R22,=1,0
1808 00
1809               NOKEY    BSZ 0
1809 52 1A E7               PUMD R22,-R32             ! PUSH FLAG ON STACK
180C 9E                     RTN
180D               !
```

```
                                                                     70
                   !          ' FORGET
180D C1            L700     BYT 301
180E A7                     ASP "'"                   ! '
180F 66 17                  DEF L517
1811 B8 05         TICK     DEF DOCOL
1813 A5 0D                  DEF DFIND
1815 59 04                  DEF ZEQU
1817 23 06                  DEF ZERO
1819 75 09                  DEF QERR
181B 03 05                  DEF DROP
181D C2 0E                  DEF LITER
181F 12 04                  DEF SEMIS
1821 86            L701     BYT 206
1822 46 4F 52               ASP "FORGET"              ! FORGET
1825 47 45 D4
1828 0D 18                  DEF L700
182A B8 05         FORGE    DEF DOCOL
182C A0 07                  DEF CURR
182E 5D 05                  DEF AT
1830 92 07                  DEF CONT
1832 5D 05                  DEF AT
1834 4B 08                  DEF SUB
1836 83 00                  DEF LIT
1838 18 00                  BYT 30,0
183A 75 09                  DEF QERR
183C 11 18                  DEF TICK
183E 25 05                  DEF DUP
1840 25 05                  DEF DUP
1842 83 00                  DEF LIT
1844 7C 1C                  DEF XDP
1846 F6 0E                  DEF ULESS
1848 33 04                  DEF TOR
184A C0 06                  DEF FENCE
184C 5D 05                  DEF AT
184E F6 0E                  DEF ULESS
1850 40 04                  DEF FROMR
1852 7A 03                  DEF OR
1854 83 00                  DEF LIT
1856 15 00                  BYT 25,0
1858 75 09                  DEF QERR
185A 25 05                  DEF DUP
185C 37 09                  DEF NFA
185E C9 06                  DEF DP
1860 81 05                  DEF STORE
1862 19 09                  DEF LFA
1864 5D 05                  DEF AT
1866 92 07                  DEF CONT
1868 5D 05                  DEF AT
186A 81 05                  DEF STORE
186C 12 04                  DEF SEMIS
186E               !
```

```
                                                                     71
                   !          BACK BEGIN ENDIF THEN
186E 84            L702     BYT 204
186F 42 41 43               ASP "BACK"                ! BACK
1872 CB
1873 21 18                  DEF L701
1875 B8 05         BACK     DEF DOCOL
1877 0E 08                  DEF HERE
1879 4B 08                  DEF SUB
187B 2A 08                  DEF COMMA
187D 12 04                  DEF SEMIS
187F C5            L703     BYT 305
1880 42 45 47               ASP "BEGIN"               ! BEGIN
1883 49 CE
1885 6E 18                  DEF L702
1887 B8 05         BEGIN    DEF DOCOL
1889 8F 09                  DEF QCOMP
188B 0E 08                  DEF HERE
188D 2B 06                  DEF ONE
188F 12 04                  DEF SEMIS
1891 C5            L704     BYT 305
1892 45 4E 44               ASP "ENDIF"               ! ENDIF
1895 49 C6
1897 7F 18                  DEF L703
1899 B8 05         ENDIF    DEF DOCOL
189B 8F 09                  DEF QCOMP
189D 33 06                  DEF TWO
189F BE 09                  DEF QPAIR
18A1 0E 08                  DEF HERE
18A3 F3 04                  DEF OVER
18A5 4B 08                  DEF SUB
18A7 10 05                  DEF SWAP
18A9 81 05                  DEF STORE
18AB 12 04                  DEF SEMIS
18AD C4            L705     BYT 304
18AE 54 48 45               ASP "THEN"                ! THEN
18B1 CE
18B2 91 18                  DEF L704
18B4 B8 05         THEN     DEF DOCOL
18B6 99 18                  DEF ENDIF
18B8 12 04                  DEF SEMIS
18BA C2            L706     BYT 302
18BB 44 CF                  ASP "DO"
18BD AD 18                  DEF L705
18BF B8 05         DO       DEF DOCOL
18C1 08 0A                  DEF COMP
18C3 4C 01                  DEF XDO
18C5 0E 08                  DEF HERE
18C7 83 00                  DEF LIT
18C9 03 00                  BYT 3,0
18CB 12 04                  DEF SEMIS
18CD               !
```

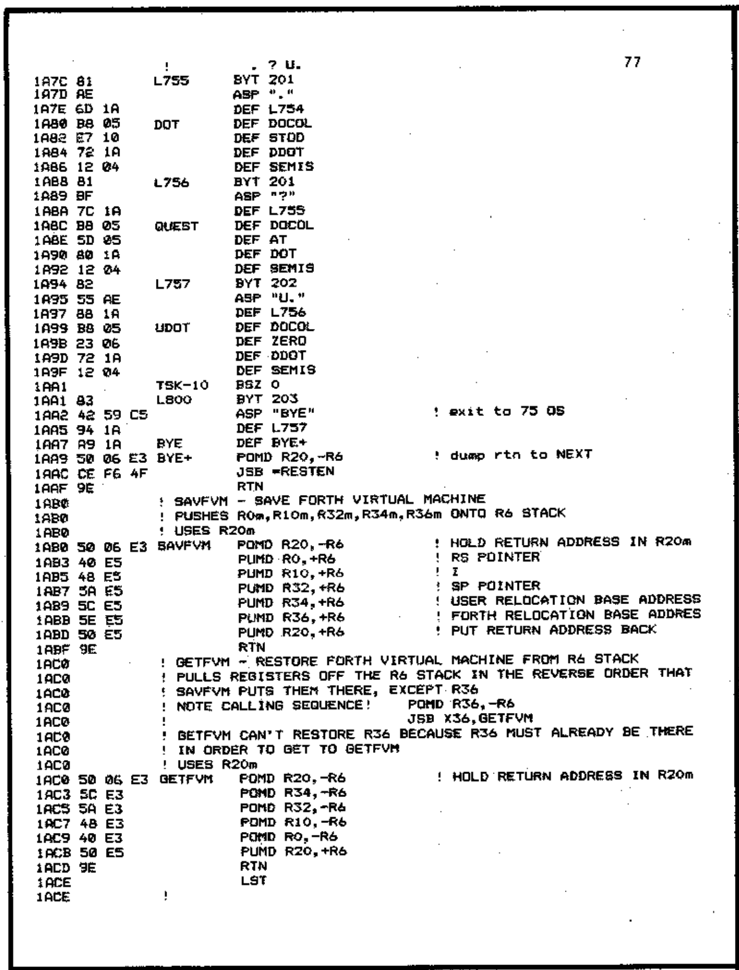
```
                                                                     72
                   !          LOOP +LOOP UNTIL END
18CD C4            L707     BYT 304
18CE 4C 4F 4F               ASP "LOOP"
18D1 D0
18D2 BA 18                  DEF L706
18D4 B8 05         LOOP     DEF DOCOL
18D6 83 00                  DEF LIT
18D8 03 00                  BYT 3,0
18DA BE 09                  DEF QPAIR
18DC 08 0A                  DEF COMP
18DE F3 00                  DEF XLOOP
18E0 75 18                  DEF BACK
18E2 12 04                  DEF SEMIS
18E4 C5            L708     BYT 305
18E5 2B 4C 4F               ASP "+LOOP"               ! +LOOP
18E8 4F D0
18EA CD 18                  DEF L707
18EC B8 05         PLOOP    DEF DOCOL
18EE 83 00                  DEF LIT
18F0 03 00                  BYT 3,0
18F2 BE 09                  DEF QPAIR
18F4 08 0A                  DEF COMP
18F6 12 01                  DEF XPLOO
18F8 75 18                  DEF BACK
18FA 12 04                  DEF SEMIS
18FC C5            L709     BYT 305
18FD 55 4E 54               ASP "UNTIL"               ! UNTIL
1900 49 CC
1902 E4 18                  DEF L708
1904 B8 05         UNTIL    DEF DOCOL
1906 2B 06                  DEF ONE
1908 BE 09                  DEF QPAIR
190A 08 0A                  DEF COMP
190C DE 00                  DEF ZBRAN
190E 75 18                  DEF BACK
1910 12 04                  DEF SEMIS
1912 C3            L710     BYT 303
1913 45 4E C4               ASP "END"                 ! END
1916 FC 18                  DEF L709
1918 B8 05         END      DEF DOCOL
191A 04 19                  DEF UNTIL
191C 12 04                  DEF SEMIS
191E               !
```

```
                     !          AGAIN REPEAT IF ELSE                        73
191E C5          L711      BYT 305
191F 41 47 41              ASP "AGAIN"               ! AGAIN
1922 49 CE
1924 12 19                 DEF L710
1926 B8 05       AGAIN     DEF DOCOL
1928 2B 06                 DEF ONE
192A BE 09                 DEF QPAIR
192C 08 0A                 DEF COMP
192E B1 00                 DEF BRAN
1930 75 18                 DEF BACK
1932 12 04                 DEF SEMIS
1934 C6          L712      BYT 306
1935 52 45 50              ASP "REPEAT"              ! REPEAT
1938 45 41 D4
193B 1E 19                 DEF L711
193D B8 05       REPEAT    DEF DOCOL
193F 33 04                 DEF TOR
1941 33 04                 DEF TOR
1943 26 19                 DEF AGAIN
1945 40 04                 DEF FROMR
1947 40 04                 DEF FROMR
1949 33 06                 DEF TWO
194B 4B 08                 DEF SUB
194D 99 18                 DEF ENDIF
194F 12 04                 DEF SEMIS
1951 C2          L713      BYT 302
1952 49 C6                 ASP "IF"                  ! IF
1954 34 19                 DEF L712
1956 B8 05       IF        DEF DOCOL
1958 08 0A                 DEF COMP
195A DE 00                 DEF ZBRAN
195C 0E 08                 DEF HERE
195E 23 06                 DEF ZERO
1960 2A 08                 DEF COMMA
1962 33 06                 DEF TWO
1964 12 04                 DEF SEMIS
1966 C4          L714      BYT 304
1967 45 4C 53              ASP "ELSE"                ! ELSE
196A C5
196B 51 19                 DEF L713
196D B8 05       ELSE      DEF DOCOL
196F 33 06                 DEF TWO
1971 BE 09                 DEF QPAIR
1973 08 0A                 DEF COMP
1975 B1 00                 DEF BRAN
1977 0E 08                 DEF HERE
1979 23 06                 DEF ZERO
197B 2A 08                 DEF COMMA
197D 10 05                 DEF SWAP
197F 33 06                 DEF TWO
1981 99 18                 DEF ENDIF
1983 33 06                 DEF TWO
1985 12 04                 DEF SEMIS
1987             !
```

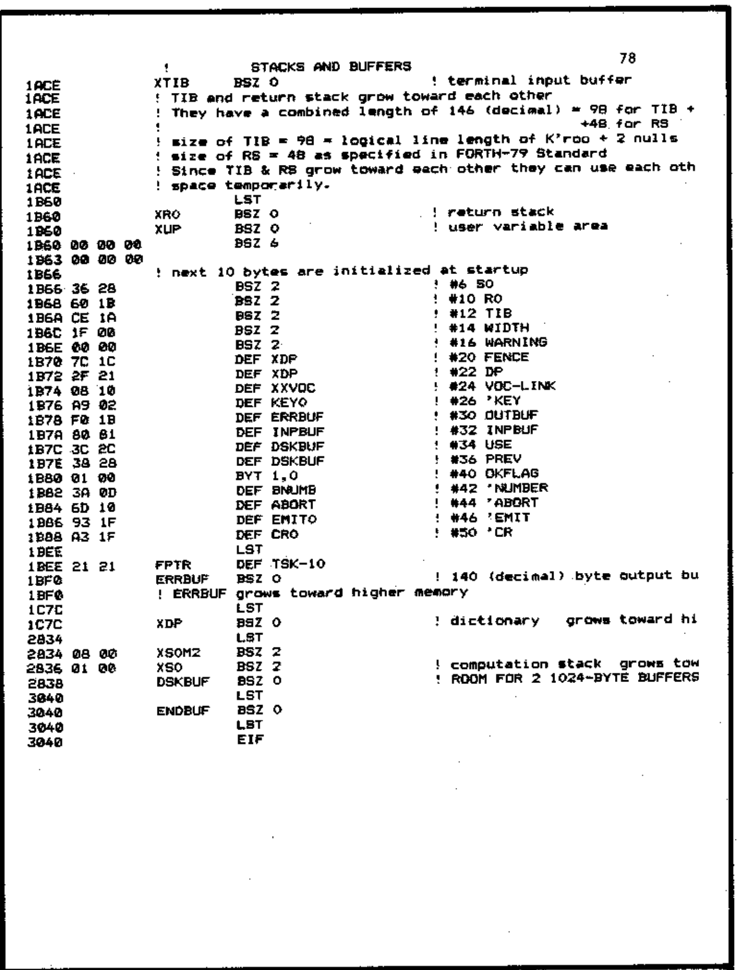
```
                     !          WHILE SPACES                                74
1987 C5          L715      BYT 305
1988 57 48 49              ASP "WHILE"               ! WHILE
198B 4C C5
198D 66 19                 DEF L714
198F B8 05       WHILE     DEF DOCOL
1991 56 19                 DEF IF
1993 FE 07                 DEF TWOP
1995 12 04                 DEF SEMIS
1997 86          L716      BYT 206
1998 53 50 41              ASP "SPACES"              ! SPACES
199B 43 45 D3
199E 87 19                 DEF L715
19A0 B8 05       SPACS     DEF DOCOL
19A2 23 06                 DEF ZERO
19A4 3D 11                 DEF MAX
19A6 D0 08                 DEF DDUP
19A8 DE 00                 DEF ZBRAN
19AA 0C 00                 BYT 14,0
19AC 23 06                 DEF ZERO
19AE 4C 01                 DEF XDO
19B0 BF 08       XXRA      DEF SPACE
19B2 F3 00                 DEF XLOOP
19B4 FC FF                 BYT 374,377
19B6 12 04       XXR4      DEF SEMIS
19B8 82          L717      BYT 202
19B9 3C A3                 ASP "<#"                  ! <#
19BB 97 19                 DEF L716
19BD B8 05       BDIGS     DEF DOCOL
19BF 89 0C                 DEF PAD
19C1 E8 07                 DEF HLD
19C3 81 05                 DEF STORE
19C5 12 04                 DEF SEMIS
19C7 82          L718      BYT 202
19C8 23 BE                 ASP "#>"                  ! #>
19CA B8 19                 DEF L717
19CC B8 05       EDIGS     DEF DOCOL
19CE 03 05                 DEF DROP
19D0 03 05                 DEF DROP
19D2 E8 07                 DEF HLD
19D4 5D 05                 DEF AT
19D6 89 0C                 DEF PAD
19D8 F3 04                 DEF OVER
19DA 4B 08                 DEF SUB
19DC 12 04                 DEF SEMIS
19DE 84          L719      BYT 204
19DF 53 49 47              ASP "SIGN"                ! SIGN
19E2 CE
19E3 C7 19                 DEF L718
19E5 B8 05       SIGN      DEF DOCOL
19E7 A2 08                 DEF ROT
19E9 6F 04                 DEF ZLESS
19EB DE 00                 DEF ZBRAN
19ED 08 00                 BYT 10,0                  ! XXR1
19EF 83 00                 DEF LIT
19F1 2D 00                 BYT 55,0
19F3 71 0C                 DEF HOLD
19F5 12 04       XXR1      DEF SEMIS
19F7             !
```

```
                     !          # #S                                        75
19F7 81          L750      BYT 201
19F8 A3                    ASP "#"
19F9 DE 19                 DEF L719
19FB B8 05       DIG       DEF DOCOL
19FD B7 07                 DEF BASE
19FF 5D 05                 DEF AT
1A01 51 12                 DEF MSMOD
1A03 A2 08                 DEF ROT
1A05 83 00                 DEF LIT
1A07 09 00                 BYT 11,0
1A09 F3 04                 DEF OVER
1A0B 6F 08                 DEF LESS
1A0D DE 00                 DEF ZBRAN
1A0F 08 00                 BYT 10,0                  ! XXR2
1A11 83 00                 DEF LIT
1A13 07 00                 BYT 7,0
1A15 84 04                 DEF PLUS
1A17 83 00       XXR2      DEF LIT
1A19 30 00                 BYT 60,0
1A1B 84 04                 DEF PLUS
1A1D 71 0C                 DEF HOLD
1A1F 12 04                 DEF SEMIS
1A21 82          L751      BYT 202
1A22 23 D3                 ASP "#S"
1A24 F7 19                 DEF L750
1A26 B8 05       DIGS      DEF DOCOL
1A28 FB 19       XXR3      DEF DIG
1A2A F3 04                 DEF OVER
1A2C F3 04                 DEF OVER
1A2E 7A 03                 DEF OR
1A30 59 04                 DEF ZEQU
1A32 DE 00                 DEF ZBRAN
1A34 F4 FF                 BYT 364,377               ! XXR3
1A36 12 04                 DEF SEMIS
1A38             !
```

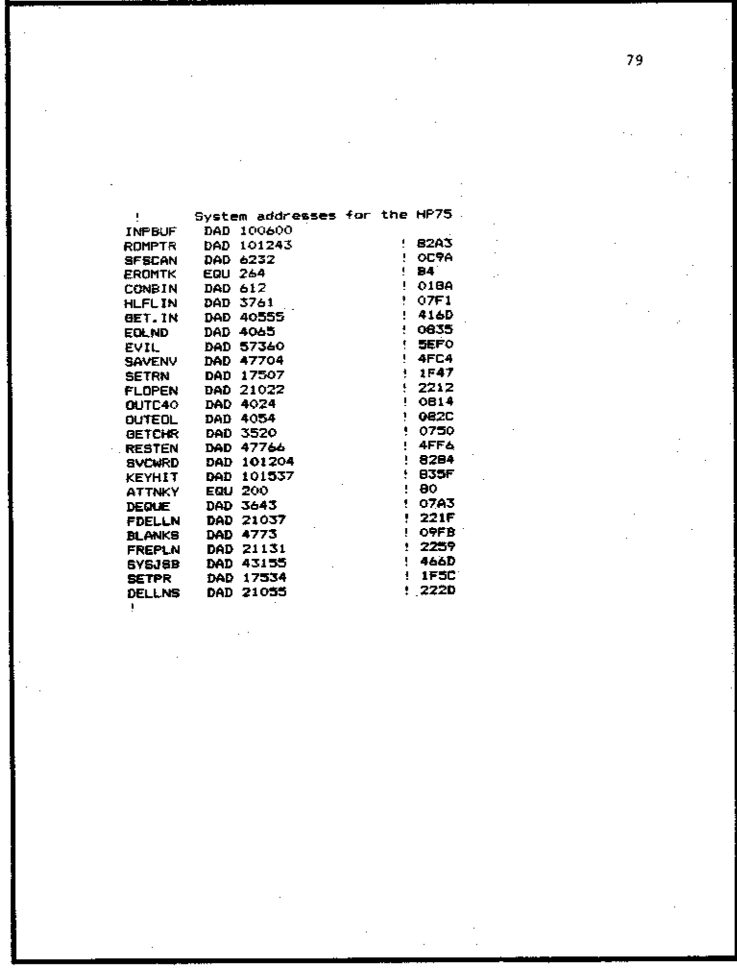
```
                     !          D.R .R D.                                   76
1A38 83          L752      BYT 203
1A39 44 2E D2              ASP "D.R"
1A3C 21 1A                 DEF L751
1A3E B8 05       DDOTR     DEF DOCOL
1A40 33 04                 DEF TOR
1A42 10 05                 DEF SWAP
1A44 F3 04                 DEF OVER
1A46 11 11                 DEF DABS
1A48 BD 19                 DEF BDIGS
1A4A 26 1A                 DEF DIGS
1A4C E5 19                 DEF SIGN
1A4E CC 19                 DEF EDIGS
1A50 40 04                 DEF FROMR
1A52 F3 04                 DEF OVER
1A54 4B 08                 DEF SUB
1A56 A0 19                 DEF SPACS
1A58 F8 0A                 DEF TYPE
1A5A 12 04                 DEF SEMIS
1A5C 82          L753      BYT 202
1A5D 2E D2                 ASP ".R"
1A5F 38 1A                 DEF L752
1A61 B8 05       DOTR      DEF DOCOL
1A63 33 04                 DEF TOR
1A65 E7 10                 DEF STOD
1A67 40 04                 DEF FROMR
1A69 3E 1A                 DEF DDOTR
1A6B 12 04                 DEF SEMIS
1A6D 82          L754      BYT 202
1A6E 44 AE                 ASP "D."
1A70 5C 1A                 DEF L753
1A72 B8 05       DDOT      DEF DOCOL
1A74 23 06                 DEF ZERO
1A76 3E 1A                 DEF DDOTR
1A78 BF 08                 DEF SPACE
1A7A 12 04                 DEF SEMIS
1A7C             !
```



```
                      !          . ? U.                                  77
1A7C 81          L755      BYT 201
1A7D AE                    ASP "."
1A7E 6D 1A                 DEF L754
1A80 B8 05       DOT       DEF DOCOL
1A82 E7 10                 DEF STOD
1A84 72 1A                 DEF DDOT
1A86 12 04                 DEF SEMIS
1A88 81          L756      BYT 201
1A89 BF                    ASP "?"
1A8A 7C 1A                 DEF L755
1A8C B8 05       QUEST     DEF DOCOL
1A8E 5D 05                 DEF AT
1A90 80 1A                 DEF DOT
1A92 12 04                 DEF SEMIS
1A94 82          L757      BYT 202
1A95 55 AE                 ASP "U."
1A97 88 1A                 DEF L756
1A99 B8 05       UDOT      DEF DOCOL
1A9B 23 06                 DEF ZERO
1A9D 72 1A                 DEF DDOT
1A9F 12 04                 DEF SEMIS
1AA1             TSK-10    BSZ 0
1AA1 83          L800      BYT 203
1AA2 42 59 C5              ASP "BYE"                 ! exit to 75 OS
1AA5 94 1A                 DEF L757
1AA7 A9 1A       BYE       DEF BYE+
1AA9 50 06 E3 BYE+         POMD R20,-R6              ! dump rtn to NEXT
1AAC CE F6 4F              JSB =RESTEN
1AAF 9E                    RTN
1AB0             ! SAVFVM - SAVE FORTH VIRTUAL MACHINE
1AB0             ! PUSHES R0m,R10m,R32m,R34m,R36m ONTO R6 STACK
1AB0             ! USES R20m
1AB0 50 06 E3 SAVFVM       POMD R20,-R6              ! HOLD RETURN ADDRESS IN R20m
1AB3 40 E5                 PUMD R0,+R6               ! RS POINTER
1AB5 48 E5                 PUMD R10,+R6              ! I
1AB7 5A E5                 PUMD R32,+R6              ! SP POINTER
1AB9 5C E5                 PUMD R34,+R6              ! USER RELOCATION BASE ADDRESS
1ABB 5E E5                 PUMD R36,+R6              ! FORTH RELOCATION BASE ADDRES
1ABD 50 E5                 PUMD R20,+R6              ! PUT RETURN ADDRESS BACK
1ABF 9E                    RTN
1AC0             ! GETFVM - RESTORE FORTH VIRTUAL MACHINE FROM R6 STACK
1AC0             ! PULLS REGISTERS OFF THE R6 STACK IN THE REVERSE ORDER THAT
1AC0             ! SAVFVM PUTS THEM THERE, EXCEPT R36
1AC0             ! NOTE CALLING SEQUENCE!     POMD R36,-R6
1AC0             !                            JSB X36,GETFVM
1AC0             ! GETFVM CAN'T RESTORE R36 BECAUSE R36 MUST ALREADY BE THERE
1AC0             ! IN ORDER TO GET TO GETFVM
1AC0             ! USES R20m
1AC0 50 06 E3 GETFVM       POMD R20,-R6              ! HOLD RETURN ADDRESS IN R20m
1AC3 5C E3                 POMD R34,-R6
1AC5 5A E3                 POMD R32,-R6
1AC7 48 E3                 POMD R10,-R6
1AC9 40 E3                 POMD R0,-R6
1ACB 50 E5                 PUMD R20,+R6
1ACD 9E                    RTN
1ACE                       LST
1ACE              !
```



```
                   !        STACKS AND BUFFERS                            78
1ACE             XTIB      BSZ 0                   ! terminal input buffer
1ACE             ! TIB and return stack grow toward each other
1ACE             ! They have a combined length of 146 (decimal) = 98 for TIB +
1ACE             !                                                  +48 for RS
1ACE             ! size of TIB = 98 = logical line length of K'roo + 2 nulls
1ACE             ! size of RS = 48 as specified in FORTH-79 Standard
1ACE             ! Since TIB & RS grow toward each other they can use each oth
1ACE             ! space temporarily.
1B60                       LST
1B60             XR0       BSZ 0                   ! return stack
1B60             XUP       BSZ 0                   ! user variable area
1B60 00 00 00              BSZ 6
1B63 00 00 00
1B66             ! next 10 bytes are initialized at startup
1B66 36 28                 BSZ 2                   ! #6 S0
1B68 60 1B                 BSZ 2                   ! #10 R0
1B6A CE 1A                 BSZ 2                   ! #12 TIB
1B6C 1F 00                 BSZ 2                   ! #14 WIDTH
1B6E 00 00                 BSZ 2                   ! #16 WARNING
1B70 7C 1C                 DEF XDP                 ! #20 FENCE
1B72 2F 21                 DEF XDP                 ! #22 DP
1B74 08 10                 DEF XXVOC               ! #24 VOC-LINK
1B76 A9 02                 DEF KEYO                ! #26 'KEY
1B78 F0 1B                 DEF ERRBUF              ! #30 OUTBUF
1B7A 80 81                 DEF INPBUF              ! #32 INPBUF
1B7C 3C 2C                 DEF DSKBUF              ! #34 USE
1B7E 38 28                 DEF DSKBUF              ! #36 PREV
1B80 01 00                 BYT 1,0                 ! #40 OKFLAG
1B82 3A 0D                 DEF BNUMB               ! #42 'NUMBER
1B84 6D 10                 DEF ABORT               ! #44 'ABORT
1B86 93 1F                 DEF EMITO               ! #46 'EMIT
1B88 A3 1F                 DEF CRO                 ! #50 'CR
1BEE                       LST
1BEE 21 21       FPTR      DEF TSK-10
1BF0             ERRBUF    BSZ 0                   ! 140 (decimal) byte output bu
1BF0             ! ERRBUF grows toward higher memory
1C7C                       LST
1C7C             XDP       BSZ 0                   ! dictionary   grows toward hi
2834                       LST
2834 08 00       XSOM2     BSZ 2
2836 01 00       XS0       BSZ 2                   ! computation stack  grows tow
2838             DSKBUF    BSZ 0                   ! ROOM FOR 2 1024-BYTE BUFFERS
3040                       LST
3040             ENDBUF    BSZ 0
3040                       LST
3040                       EIF
```



```
                                                                          79

 !        System addresses for the HP75
INPBUF    DAD 100600
ROMPTR    DAD 101243              ! 82A3
SFSCAN    DAD 6232                ! 0C9A
EROMTK    EQU 264                 ! B4
CONBIN    DAD 612                 ! 018A
HLFLIN    DAD 3761                ! 07F1
GET.IN    DAD 40555               ! 416D
EOLND     DAD 4065                ! 0835
EVIL      DAD 57360               ! 5EF0
SAVENV    DAD 47704               ! 4FC4
SETRN     DAD 17507               ! 1F47
FLOPEN    DAD 21022               ! 2212
OUTC40    DAD 4024                ! 0814
OUTEOL    DAD 4054                ! 082C
GETCHR    DAD 3520                ! 0750
RESTEN    DAD 47766               ! 4FF6
SVCWRD    DAD 101204              ! 8284
KEYHIT    DAD 101537              ! 835F
ATTNKY    EQU 200                 ! 80
DEQUE     DAD 3643                ! 07A3
FDELLN    DAD 21037               ! 221F
BLANKS    DAD 4773                ! 09FB
FREPLN    DAD 21131               ! 2259
SYSJSB    DAD 43155               ! 466D
SETPR     DAD 17534               ! 1F5C
DELLNS    DAD 21055               ! 222D
 !
```

KERNAL GLOSSARY

This glossary contains all of the word definitions in the assembly source listing. The definitions are presented in the order of their ascii sort.

The first line of each entry shows a symbolic description of the action of the procedure on the parameter stack. The symbols indicate the order in which input parameters have been placed on the stack. Three dashes "---" indicate the execution point; any parameters left on the stack are listed. In this notation, the top of the stack is to the right.

The symbols include:

addr	memory address
b	8 bit byte (i.e. hi 8 bits zero)
c	7 bit ascii character (hi 9 bits zero)
d	32 bit signed double integer, most significant portion with sign on top of stack.
f	boolean flag. 0=false, non-zero=true
ff	boolean false flag=0
n	16 bit signed integer number
u	16 bit unsigned integer
tf	boolean true flag=non-zero

Unless otherwise noted, all reference to numbers are for 16 bit signed integers. For 32 bit signed double numbers, the most significant part (with the sign) is on top.

All arithemetic is implicity 16 bit signed integer math, with error and under-flow indication unspecified.

Source
Listing
Page

!   n addr --- (relative address)   17
Store 16 bits of n at address. Pronounced "store".

!CSP   28
Save the stack position in CSP, as part of the compiler security.

d1 --- d2   75
Generate from a double number d1, the next ascii character which is placed in an output string. Result d2 is the quotient after division by BASE, and is maintained for further processing. Used between <# and #>. See #S.

#>   d --- addr count   74
Terminiates numeric output conversion by dropping d, leaving the test address and character count suitable for TYPE.

#S   d1 --- d2   75
Generates ascii text in the text output buffer, by the use of #, until a zero double number n2 results. Used between <# and #>.

'   --- addr   75
Used in the form: ' nnnn Leaves the parameter field address of dictionary word nnnn. As a compiler directive, executes in a colon-definition to compile the address as a literal. If the word is not found after a search of CONTEXT and CURRENT, an appropriate error message is given. Pronounced "tick".

'ABORT   --- addr   22
A user variable containing the CFA of the active (ABORT) routine. Initialized to execute ABORT. See ABORT, (ABORT).

'CR   --- addr   22
A user variable containing the CFA of the active CR routine. Initialized to execute CRØ. See CR CRØ. Change to redirect character output.

'EMIT   --- addr   22
A user variable containing the CFA of the active EMIT routine. Initialized to execute EMITØ. See EMIT EMITØ. Change to redirect character output.

'KEY   --- addr   21
A user variable containing the CFA of the active KEY routine. Initialized to execute KEYØ. See KEY KEYØ. Change to accept characters from a different source than keyboard, i.e. modern.

'NUMBER   --- addr   22
A user variable containing the CFA of the active NUMBER conversion routine. Initialized to execute [NUMBER]. See [NUMBER] NUMBER. Change to modify numeric processing, e.g. to accept scientific notation.

(47
Used in the form: (cccc) Ignore a comment that will be delimited by a right parenthesis on the same line. May occur during execution or in a colon-definition. A blank after the leading parenthesis is required.

(.")   33
The run-time procedure, compiled by ." which transmits the following in-line text to the selected output device. See .".

(;CODE)   30
The run-time procedure, compiled by ;CODE, that rewrites the code field of the most recently defined word to point to the following machine code sequence. See ;CODE.

(+LOOP)   n ---   5
The run-time procedure compiled by +LOOP, which increments the loop index by n and tests for loop completion. See +LOOP.

(ABORT)   40
Executes after an error when WARNING is 1. This word normally executes ABORT, but may be altered to a user's alternative procedure. See 'ABORT, ABORT.

(DO)   5
The run-time procedure compiled by DO which moves the loop control parameters to the return stack. See DO.

(FIND)   addr1 addr2 --- pfa b tf (ok)   7
addr1 addr2 --- ff (bad)
Searches the dictionary starting at the name field address addr2, matching to the text at addr1. Returns parameter field address, length byte of name field and boolean true for a match. If no match is found, only a boolean false is left.

(LINE)   n1 n2 --- addr count   59
Convert the line number n1 and the screen n2 to the disc buffer address containing the data. a count of LC/L indicates the full line text length.

(LOOP) 4
The run-time procedure compiled by LOOP which increments the loop index and tests for loop completion. See LOOP.

(NUMBER) d1 addr1 --- d2 addr2 38
Convert the ascii text beginning at addr1+1 with regard to BASE. The new value is accumulated into double number d1, being left as d2. Addr2 is the address of the first unconvertible digit. Used by NUMBER and [NUMBER].

* n1 n2 --- prod 53
Leave the signed product of two signed numbers.

*/ N1 n2 n3 --- n4 54
Leave the ratio n4 = n1*n2/n3 where all are signed numbers. Retention of an intermediate 31 bit product permits greater accuracy than would be available with the sequence:
n1 n2 * n3 /

*/MOD n1 n2 n3 --- n4 n5 54
Leave the quotient N5 and remainder n4 of the operation n1*n2/n3 A 31 bit intermediate product is used as for */.

+ n1 n2 --- sum 15
Leave the sum of n1*n2.

+! n addr --- 16
Add n to the value at the address. Pronounced "plus-store".

+BUF add1 --- addr2 f 55
Advance the disc buffer address addr1 to the address of the next buffer addr2. Boolean f is false when addr2 is the buffer presently pointed to by variable PREV.

+LOOP n1 --- (run) 72
addr N2 --- (compile)
Used in a colon-definition in the form:
DO ... n1 +LOOP
At run-time, +LOOP selectively controls branching back to the corresponding DO based on n1, the loop index and the loop limit. The signed increment n1 is added to the index and the total compared to the limit. The branch back to DO occurs until the new index is equal or greater than the limit (n1>0), or until the new index is equal to or less than the limit (n1< 0). Upon exiting the loop, the parameters are discarded and execution continues ahead.

At compile time, +LOOP compiles the run-time word (+LOOP) and the branch offset computed from HERE to the address left on the stack by DO. n2 is used for compile time error checking.

, n --- 24
Store n into the next available dictionary memory cell, advancing the dictionary pointer. (comma)

- n1 n2 --- diff 25
Leave the difference of n1-n2.

--> 63
Continue interpretation with the next disc screen. (pronounced next-screen).

-DUP n1 -- n1 (if zero) 26
n1 -- n1 n1 (non-zero)
Reproduce n1 only if it is non-zero. This is usually used to copy a value just before IF, to eliminate the need for an ELSE part to drop it.

-FIND --- pfa b tf (found 40
--- ff (not found)
Accepts the next test word (delimited by blanks) in the input stream to HERE, and searches the CONTEXT and then CURRENT vocabularies for a matching entry. If found, the dictionary entry's parameter field address, its length byte, and a boolean true is left. Otherwise, only a boolean false is left.

-TRAILING addr n1 --- addr n2 32
Adjusts the character count n1 of a text string beginning address to suppress the output of trailing blanks. i.e. the characters at addr+n1 to addr+n2 are blanks.

. n --- 77
Print a number from a signed 16 bit two's complement value, converted according to the numeric BASE. A trailing blank follows. Pronounced "dot".

." Used in the form: ." cccc" 33
Compiles in in-line string cccc (delimited by the trailing ") with an execution procedure to transmit the text to the selected output device. If executed outside a definition, ." will immediately print the text until the final ". The maximum number of characters is 255. See (.").

.LINE line scr --- 59
Print on the terminal device, a line of text from a screen by its line and screen number. Trailing blanks are suppressed.

.R n1 n2 --- 76
Print the number n1 right aligned in a field whose width is n2. No following blank is printed.

/ n1 n2 --- quot 53
Leave the signed quotient of n1/n2.

/MOD n1 n2 --- rem quot 53
Leave the remainder and signed quotient of n1/n2. The remainder has the sign of the dividend.

0 1 2 3 --- m 19
These small numbers are used so often that it is attractive to define them by name in the dictionary as consants.

0< n --- f 14
Leave a true flag if the number is less than zero (negative), otherwise leave a false flag.

0= n --- f 14
Leave a true flag is the number is equal to zero, otherwise leave a false flag.

0BRANCH f --- 4
The run-time procedure to conditionally branch. If f is false (zero), the following in-line parameter is added to the interpretive pointer to branch ahead or back. Compiled by IF, UNTIL, and WHILE.

1+ n1 --- n2 24
Increment n1 by 1

2+ n1 --- n2 24
Increment n1 by 2.

:   18

Used in the form called a colon-definition:

: cccc ... ;

Creates a dictionary entry defining cccc as equivalent to the following sequence of Forth word definitions '...' until the next ';' or ';CODE'. The compiling process is done by the text interpreter as long as STATE is non-zero. Other details are that the CONTEXT vocabulary is set to the CURRENT vocabulary and that words with the precedence bit set (P) are executed rather than being compiled.

;   18

Terminate a colon-definition and stop further compilation. Compiles the run-time ;S.

;CODE   See 75C Assembler   Screen 30

;S   13

Stop interpretation of a screen. ;S is also the run-time word compiled at the end of a colon-definition which returns execution to the calling procedure.

<   n1 n2 --- f   25

Leave a true flag if n1 is less than n2; otherwise leave a false flag. This is a signed comparison. See U .

<#   74

Setup for pictured numeric output formation using the words: <# # #S SIGN #>. The conversion is done on a double number producing text at PAD.

<<   n1 --- n2   12

Logical left shift one bit position. Can be used for 2*.

<BUILDS   31

Used within a colon-definition:

: cccc <BUILDS ... DOES> ...;

Each time cccc is executed, <BUILDS defines a new word with a high-level execution procedure. Executing cccc in the form: cccc nnnn used <BUILDS to create a dictionary entry for nnnn with a call to the DOES> part for nnnn. When nnnn is later executed, it has the address of its parameter area on the stack and executes the words after DOES> in cccc. <BUILDS and DOES> allow run-time procedures to written in high-level rather than in assembler code (as required by ;CODE).

=   n1 n2 --- f   25

Leave a true flag if n1=n2; otherwise leave a false flag.

>   n1 n2 --- f   25

Leave a true flag if n1 is greater than n2; otherwise a false flag. This is a signed comparison.

>>   n1 --- n2   12

Logical right shift one bit position. Can be used for 2/.

>R   n ---   14

Remove a number from the computation stack and place as the most accessable on the return stack. Use should be balanced with R> in the same definition.

?   addr ---   77

Print the value contained at the address in free format according to the current base.

?COMP   28

Issue error message if not compiling.

?CSP   29

Issue error message if stack position differs from value saved in CSP.

?ERROR   f n ---   28

Issue an error message number n, if the boolean flag is true.

?EXEC   28

Issue an error message if not executing.

?LOADING   29

Issue an error message if not loading.

?PAIRS   n1 n2 ---   28

Issue an error message if n1 does not equal n2. The message indicates that compiled conditionals do not match.

?STACK   44

Issue an error message if the stack is out of bounds.

?TERMINAL   --- f   9

Perform a test of the terminal keyboard for actuation of a key. A true flag indicates actuation.

@   addr --- n (relative address)   17

Leave the 16 bit contents of address.

A>>   n1 --- n2   12

Arithmetic right shift one bit position.

ABORT   48

Clear the stacks and enter the execution state. Return control to the operators terminal, printing a message appropriate to the installation.

ABS   n --- u   50

Leave the absolute value of n as u.

AGAIN   addr n --- (compiling)   73

Used in a colon-definition in the form:

BEGIN ... AGAIN.

At run-time, AGAIN forces execution to return to corresponding BEGIN. There is no effect on the stack. Execution cannot leave this loop (unless R> DROP is executed one level below).

At compile time, AGAIN compiles BRANCH with an offset from HERE to addr. n is used for compile-time error checking.

ALLOT   n ---   24

Add the signed number to the dictionary pointer DP. May be used to reserve dictionary space or re-origin memory. n is with regard to computer address type (byte or word).

AND   n1 n2 --- n2   11

Leave the bitwise logical and of n1 and n2 as n3.

B/BUF   --- n   20

This constant leaves the number of bytes per buffer.

B/SCR   --- n   20

This constant leaves the number of blocks per editing screen.

BACK   addr ---   71

Calculate the backward branch offset from HERE to addr and compile into the next available dictionary memory address.

BASE --- addr 23
A user variable containing the current number base used for input and output conversion.

BEGIN --- addr n (compiling) 70
Occurs in a colon definition in form:

```
BEGIN ... UNTIL
BEGIN ... AGAIN
BEGIN ... WHILE ... REPEAT
```

At run-time, BEGIN marks the start of a sequence that may be repetitively executed. It serves as a return point from the corresponding UNTIL, AGAIN or REPEAT. When executing UNTIL, a return to BEGIN will occur if the top of the stack is false; for AGAIN and REPEAT a return to BEGIN always occurs.

At compile time BEGIN leaves its return address and n for compiler error checking.

BL --- c 19
A constant that leaves the ascii value for "blank".

BLANKS addr count --- 36
Fill an area of memory beginning at addr with blanks.

BLK --- addr 22
A user variable containing the block number being interpreted. If zero, input is being taken from the terminal input buffer.

BLOCK n --- addr 58
Leave the memory address of the block buffer containing block n. If the block is not already in memory, it is transferred from a text file to a buffer. If the block occupying that buffer has been marked as updated, it is rewritten to disc before block n is read into the buffer. See also BUFFER, R/W, UPDATE, FLUSH.

BRANCH 4
The run-time procedure to unconditionally branch. An in-line offset is added to the interpretive pointer IP to branch ahead or back. BRANCH is compiled by ELSE, AGAIN, REPEAT.

BUFFER n --- addr 57
Obtain the next memory buffer, assigning it to block n. If the contents of the buffer is marked as updated, it is written to the text file. The address left is the first cell within the buffer for data storage.

BYE 77
Return to HP-75C operating system.

C! b addr --- (relative address) 17
Store 8 bits at address.

C, b --- 24
Store 8 bits of b into the next available dictionary byte, advancing the dictionary pointer.

C/L 19
A constant containing the number of characters per line. Can be changed to 20 hex by the command 20 ' C/L !.

C@ addr --- b (relative address) 17
Leave the 8 bit contents of memory address.

CFA pfa --- cfa 27
Convert the parameter field address of a definition to its code field address.

CMOVE from to count --- (relative address) 10
Move the specified quantity of bytes beginning at address from to address to. The contents of address from is moved first proceeding toward high memory.

COM n1 --- n2 12
Does a one's complement to the item on top of the stack.

COMPILE 29
When the word containing COMPILE executes, the execution address of the word following COMPILE is copied (compiled) into the dictionary. This allows specific compilation situations to be handled in addition to simply compiling an execution address (which the interpreter already does).

CONSTANT n --- 18
A defining word used in the form:

```
n CONSTANT ccc
```

to create word cccc, with its parameter field containing n. When cccc is later executed, it will push the value of n to the stack.

CONTEXT --- addr 23
A user variable containing a pointer to the vocabulary within which dictionary searches will first begin.

COUNT addr1 --- addr2 n 31
Leave the byte address addr2 and byte count n of a message test beginning at address addr1. It is presumed that the first byte contains the text byte count and the actual text starts with the second byte. Typically COUNT is followed by TYPE.

CR 9
Transmit a carriage return and line feed to the selected output device.

CRØ 9
Outputs an end of line sequence to output device. Is vectored so it can be redefined. See CR, 'CR.

CREATE 42
A defining word used in the form:

```
CREATE cccc
```

by such words as CODE and CONSTANT to create a dictionary header for a Forth definition. The code field contains the address of the words parameter field. The new word is created in the CURRENT vocabulary.

CSP ---- addr 23
A user variable temporarily storing the stack pointer position, for compilation error checking.

CURRENT --- addr 23
A user variable containing a pointer to the vocabulary into which new definitions will be compiled.

D+ d1 d2 --- dsum 15
Leave the double number sum of two double numbers.

D. d --- 76
Print a signed double number from a 32 bit two's complement value. The high-order 16 bits are most accessible on the stack. Conversion is performed according to the current BASE. A blank follows. Pronounced D-dot.

D.R d n --- 76
Print a signed double number d right aligned in a field n characters wide.

DABS d --- ud 50
Leave the absolute value ud of a double number.

DECIMAL 30
Set the numeric conversion BASE for decimal input-output.

DEFINITIONS 47
Used in the form: cccc DEFINITIONS
Set the CURRENT vocabulary to the CONTEXT vocabulary. In the example, executing vocabulary name cccc made it the CONTEXT vocabulary and executing DEFINITIONS made both specify vocabulary cccc.

DIGIT c n1 ---- n2 tf (ok) 6
c n1 ---- ff (bad)
Converts the ascii character c (using base n1) to its binary equivalant n2, accompanied by a true flag. If the conversion is invalid, leaves only a false flag.

DLITERAL d --- d (executing 43
d --- (compiling)
If compiling, compile a stack double number into a literal. Later execution of the definition containing the literal will push it to the stack. If executing, the number will remain on the stack.

DMINUS d1 --- d2 15
Convert d1 to its double number two's complement.

DO n1 n2 --- (execute) 71
addr n --- (compile)
Occurs in a colon-definition in form:

```
DO ... LOOP
DO ... +LOOP
```

At run-time, DO begins a sequence with repetitive execution controled by a loop limit n1 and an index with initial value n2. DO removes these from the stack. Upon reaching LOOP the index is incremented by one. Until the new index equals or exceeds the limit, execution loops back to just after DO; otherwise the loop parameters are discarded and execution continues ahead. Both n1 and n2 are determined at run-time and may be the result of other operations. Within a loop "I" will copy the current value of the index to the stack. see I, LOOP, +LOOP, LEAVE.

When compiling within the colon-definition, DO compiles (DO), leaves the following address addr and n for later error checking.

DOES> 31
A word which defines the run-time action within a high-level defining word. DOES> alters the code field and first parameter of the new word to execute the sequence of compiled word addresses following DOES>. Used in combination with <BUILDS. When the DOES> part executes it begins with the address of the first parameter of the new word on the stack. This allows interpretation using this area or its contents. Typical uses include the Forth assemble, multi-dimensional arrays, and compiler generation.

DP ---- addr 21
A user variable, the dictionary pointer, which contains the address of the next free memory above the dictionary. The value may be read by HERE and altered by ALLOT.

DPL ---- ADDR 23
A user variable containing the number of digits to the right of the decimal on double integer input. It may also be used hold output column location of a decimal point, in user generated formating. The default value on single number input is -1.

DROP n --- 16
Drop the number from the stack.

DUP n --- n n 16
Duplicate the value on the stack.

ELSE addr1 n11 --- addr2 n2 73
Occurs within a colon-definition in the form:

```
IF ... ELSE ... ENDIF
```

At run-time, ELSE executes after the true part following IF. ELSE forces execution to skip over the following false part and resumes execution after the ENDIF. It has no stack effect.

At compile-time ELSE emplaces BRANCH reserving a branch offset, leaves the address addr2 and n2 for error testing. ELSE also resolves the pending forward branch from IF by calculating the offset from addr1 to HERE and storing at addr1.

EMIT c --- 9
Transmit ascii character c to the selected output device.

EMITØ c --- 9
Outputs character to output device. Is vectored so it can be redefined. See EMIT, 'EMIT.

EMPTY-BUFFERS 55
Mark all block-buffers as empty. Updated blocks are not written to the disc. This is also an initialization procedure.

ENCLOSE addr1 c --- 8
addr1 n1 n2 n3
The text scanning primitive used by WORD. From the text address addr1 and an ascii delimiting character c, first non-delimiter character n1, the offset to the first delimiter after the text n2, and the offset to the first character not included. This procedure will not process past an ascii 'null', treating it as an unconditional delimiter.

END 72
This is an 'alias' or duplicate definition for UNTIL.

ENDIF addr n --- (compile) 71
Occurs in a colon-definition in form:

```
IF ... ENDIF
IF ... ELSE ... ENDIF
```

At run-time, ENDIF serves only as the destination of a forward branch from IF or ELSE. It marks the conclusion of the conditional structure. THEN is another name for ENDIF. Both names are supported in fig-FORTH. See also IF and ELSE.

At compile-time, ENDIF computes the forward branch offset from addr to HERE and stores it at addr. n is used for error tests.

ERASE addr n --- 36
Clear a region of memory to zero from addr over n addresses.

ERROR line --- in blk 41
Execute error notification and restart of system. WARNING is first examined. If WARNING=0, n is just printed as a message. If WARNING is -1, the definition (ABORT) is executed, which executes the system ABORT, unless 'ABORT has been changed to a user defined routine. Fig-FORTH saves the contents of IN and BLK to assist in determining the location of the error. Final action is execution of QUIT.

EXECUTE addr --- 3
Execute the definition whose code field address is on the stack. The code field address is also called the compilation address.

EXPECT addr count --- 34

Transfer characters from the terminal to address, until a "return" or the count of characters have been received. Two nulls are added at the end of the text.

FENCE --- addr 21

A user variable containing an address below which FORGETting is trapped. To forget below this point the user must alter the contents of FENCE.

FILL addr quan b --- (relative address) 35

Fill memory at the address with the specified quantity of bytes b. b must be greater than two.

FIRST --- n 20

A constant that leaves the address of the first (lowest) block buffer.

FLD --- addr 23

A user variable for control of number output field width.

FLUSH 56

Causes all UPDATEd buffers to be written to their respective text files.

FORGET 70

Executed in the form: FORGET cccc
Deletes definition named cccc from the dictionary with all entries physically following it. An error message will occur if the CURRENT and context vacabularies are not the same.

FORTH 47

The name of the primary vocabulary. Execution makes FORTH the CONTEXT vocabulary. Until additional user vocabularies are defined, new user definitions become a part of FORTH. FORTH is immediate, so it will execute during the creation of a colon-definition, to select this vocabulary at compile time.

HERE --- addr 24

Leave the address of the next available dictionary location.

HEX 30

Set the numeric conversion base to sixteen (hexadecimal).

HLD --- addr 23

A user variable that holds the address of the latest character of text during numeric output conversion.

HOLD c --- 36

Used between <# and #> to insert an ascii character into a pictured numeric output string. e.g. 2E HOLD will place a decimal point.

I --- n 5

Used within a DO-LOOP to copy the loop index to the stack. See R.

ID. addr --- 41

Print a definition's name from its name field address.

IF f --- (run-time) 73
--- addr n (compile)

Occurs in a colon-definition in form:

```
IF (tp) ... ENDIF
IF (tp) ... ELSE (fp) ... ENDIF
```

At run-time, IF selects execution based on a boolean flag. If f is true (non-zero), execution skips till just after ELSE to execute the false part. After either part, execution resumes after ENDIF. ELSE and its false part are optional; if missing, false execution skips to just after ENDIF.

At compile-time IF compiles OBRANCH and reserves space for an offset at addr. addr and n are used later for resolution of the offset and error testing.

IMMEDIATE 45

Mark the most recently made definition so that when encountered at compile time, it will be executed rather than being compiled. i.e. the precedence bit in its header is set. This method allows definitions to handle unusual compiling situations, rather than build them into the fundamental compiler. The user may force compilation of an immediate definition by preceeding it with [COMPILE].

IN --- addr 22

A user variable containing the byte offset within the current input text buffer (terminal or disc) from which the next text will be accepted. WORD uses and moves the value of IN.

INPBUF --- addr 21

A user variable initialized to the system input buffer at 8180 hex.

INTERPRET 45

The outer text interpreter which sequentially executes or compiles text from the input stream (terminal or buffer) depending on STATE. If the word name cannot be found after a search of CONTEXT and then CURRENT it is converted to a number according to the current base. That also failing, an error message echoing the name with a "?" will begin. Text input will be taken according to the convention forWØRD. If a decimal point is found as part of a number, a double number value will be left. The decimal point has no other purpose than to force this action. See NUMBER.

KEY --- c 9

Leave the ascii value of the next terminal key struck.

KEYØ --- b 9

Gets byte from input device, usually keyboard. Is vectored so it can be redefined. See KEY, 'KEY.

LATEST --- addr 27

Leave the name field address of the topmost word in the CURRENT vocabulary.

LEAVE 13

Force termination of a DO-LOOP at the next opportunity by setting the loop limit equal to the current value of the index. The index itself remains unchanged, and execution proceeds normally until LOOP or +LOOP is encountered.

LFA pfa --- lfa 27

Convert the parameter field address of a dictionary definition to its link field address.

LIMIT ---- n 20

A constant leaving the address just above the highest memory available for a buffer. Usually this is the highest system memory.

LIT --- n 3

Within a colon-definition, LIT is automatically compiled before each 16 bit literal number encountered in input text. Later execution of lit causes the contents of the next dictionary address to be pushed to the stack.

LITERAL n --- (compiling) 43

If compiling, then compile the stack value n as a 16 bit literal. This definition is immediate so that it will execute during a colon-definition. The intended use is:

```
      :  xxx    [ calculate ]  LITERAL  ;
```

Compilation is suspended for the compile-time calculation of a value. Compilation is resumed and LITERAL compiles this value.

LOAD n --- 63

Begin interpretation of screen n. Loading will terminate at the end of the screen or at ;S. See ;S and -->.

LOOP addr n --- (compiling) 72

Occurs in a colon-definition in form:

```
      DO  ...  LOOP
```

At run-time, LOOP selectively controls branching back to the corresponding DO based on the loop index and limit. The loop index is incremented by one and compared to the limit. The branch back to DO occurs until the index equals or exceeds the limit; at that time, the parameters are discarded and execution continues ahead.

At compile-time, LOOP compiles (LOOP) and uses addr to calculate an offset to DO. n is used for error testing.

M* n1 n2 --- d 51

A mixed magnitude math operation which leaves the double number signed product of two signed number.

M/ d n1 --- n2 n3 52

A mixed magnitude math operator which leaves the signed remainder n2 and signed quotient n3, from a double number dividend and divisor n1. The remainder takes its sign from the dividend.

M/MOD ud1l u2 --- u3 ud4 54

An unsigned mixed magnitude math operation which leaves a double quotient ud4 and remainder u3, from a double divident ud1 and single divisor u2.

MAX n1 n2 --- max 50

Leave the greater of two numbers.

MESSAGE n --- 61

Print on the selected output device the text of message n. 24 entries are available.

MIN n1 n2 --- min 50

Leave the smaller of two numbers.

MINUS n1 --- n2 15

Leave the two's complement of a number.

MOD n1 n2 --- mod 54

Leave the remainder of n1/n2, with the same sign as n1.

MSGADR --- addr 59

An array of system messages. Addr is the first byte of the array which is a table of pointers into the message text.

NFA pfa --- nfa 27

Convert the parameter field address of a definition to its name field.

NUMBER addr --- d 39

Causes the execution of the current numeric processing routine. Initialized to [NUMBER]. See [NUMBER], (NUMBER).

OCTAL 30

Set the numeric conversion BASE for base 8 input-output.

OFFSET --- addr 22

A user variable which may contain a block offset to mass storage drives. The contents of OFFSET is added to the stack number by BLOCK. See BLOCK.

OKFLAG --- addr 22

A user variable that enables or disables the "ok" in QUIT.

OR n1 n2 -- or 11

Leave the bit-wise logical or of two 16 bit values.

OUT --- addr 22

A user variable. The user may alter and examine OUT to control display formating.

OUTBUF 21

140 decimal byte available for temporary arrays, output, formatting, etc. Not used by the fig-FORTH. May be eliminated if space is needed.

OVER n1 n2 --- n1 n2 n1 16

Copy the second stack value, placing it as the new top.

PAD --- addr 36

Leave the address of the text output buffer, which is a fixed offset above HERE.

PFA nfa --- pfa 27

Convert the name field address of a compiled definition to its parameter field address.

PREV --- addr 21

A variable containing the address of the buffer most recently referenced. The UPDATE command marks this buffer to be later written to disc.

QUERY 34

Input 96 characters of text (or until a "return") from the operators terminal. Text is positioned at the address contained in TIB with IN set to zero.

QUIT 47

Clear the return stack, stop compilation, and return control to the operators terminal. No message is given.

R --- n 14

Copy the top of the return stack to the computation stack.

R# --- addr 23

A user variable which may contain the location of an editing cursor, or other file related function.

R/W addr blk f --- 68

The fig-FORTH standard disc read-write linkage. addr specifies the source or destination block buffer, blk is the sequential number of the referenced block; and f is a flag for f=0 write and f=1 read. R/W performs the read-write from a text file and performs any error checking.

R> --- n 14

Remove the top value from the return stack and leave it on the conputation stack. See R and >R.

RO --- addr 20

A user variable containing the initial location of the return stack. Pronounced R-zero. See RP!

RDFILE 65

Primative for moving a block from a system text file in RAM to a block buffer.

REPEAT addr n --- (compiling) 73

Used within a colon-definition in the form:

BEGIN ... WHILE... REPEAT

At run-time, REPEAT forces an unconditional branch back to just after the corresponding BEGIN.

At compile-time, REPEAT compiles BRANCH and the offset from HERE to addr. n is used for error testing.

ROT n1 n2 n3 --- n2 n3 n1 26

Rotate the top three values on the stack, bringing the third to the top.

RP! 13

A computer dependent procedure to initialize the return stack pointer from user variable RO.

S->D n --- d 49

Sign extend a single number to form a double number.

SO --- addr 20

A user variable that contains the initial value for the stack pointer. Pronounced S-zero. See SP!

SCR --- addr 22

A user variable containing the screen number most recently referenced by LIST.

SCRNAME BLK# --- addr count 64

Builds a file name acceptable to HP-75C operating system. Used by mass storage routine.

SIGN n d --- d 74

Stores an ascii "-" sign just before a converted numeric output string in the text output buffer when n is negative. n is discarded but double number d is maintained. Must be used between <# and #>.

SMUDGE 30

Used during word definition to toggle the "smudge bit" in a definitions' name field. This prevents an uncompleted definition from being found during dictionary searches, until compiling is completed without error.

SP! 13

A procedure to initialize the stack pointer from SO.

SP@ --- addr 13

A procedure to return the address of the stack position to the top of the stack, as it was before SP@ was executed. (e.g. 1 2 SP@ @ ... . would type 2 2 1)

SPACE 26

Transmit an ascii blank to the output device.

SPACES n --- 74

Transmit n ascii blanks to the output device.

STATE --- addr 23

A user variable containing the compilation state. A non-zero value indicates compilation.

SWAP n1 n2 --- n2 n1 16

Exchange the top two values on the stack.

THEN 71

An alias for ENDIF.

TIB --- addr 20

A user variable containing the address of the terminal input buffer.

TOGGLE addr b --- 17

Complement the contents of addr by the bit pattern b.

TRAVERSE addr1 n --- addr2 26

Move across the name field of a fig-FORTH variable length name field. addr1 is the address of either the length byte or the last letter. If n=1, the motion is toward high memory; if n=-1, the motion is toward low memory. The addr2 resulting is address of the other end of the name.

TYPE addr count --- 32

Transmit count characters from addr to the selected output device.

U. n --- 77

Print a number unsigned, according to the current numeric BASE. A trailing blank follows.

U* u1 u2 --- ud 10

Leave the unsigned double number product of two unsigned numbers.

U/ ud u1 --- u2 u3 11

Leave the unsigned remainder u2 and unsigned quotient u3 from the unsigned double dividend ud and unsigned divisor u1.

U< u1 u2 --- f 44

An unsigned comparison. Leaves a true flag if u1 is less than u2, else false. See .

UNTIL f --- (run-time) 72
addr n --- (compile)

Occurs within a colon-definition in the form:

BEGIN ... UNTIL

At run-time, UNTIL controls the conditional branch back to the corresponding BEGIN. If f is false, execution returns to just after BEGIN: if true, execution continues ahead.

At compile-time, UNTIL compiles (OBRANCH) and an offset from HERE to addr. n is used for error tests.

UPDATE 55

Marks the most recently referenced block (pointed to by PREV) as altered. The block will subsequently be transferred automatically to its text file should its buffer be required for storage of a different block.

USE --- addr 21

A variable containing the address of the block buffer to use next, as the least recently written.

USER n --- 19

A defining word used in the form:

n USER cccc

which creates a user variable cccc. The parameter field of cccc contains n as a fixed offset relative to the user pointer register UP for this user variable. When cccc is later executed, it places the sum of its offset and the user area base address on the stack as the storage address of that particular variable.

VARIABLE 19

A defining word used in the form:

n VARIABLE cccc

When VARIABLE is executed, it creates the definition cccc with its parameter field initialized to n. When cccc is later executed, the address of its parameter field (containing n) is left on the stack, so that a fetch or store may access this location.

VOC-LINK --- addr 21

A user variable containing the address of a field in the definition of the most recently created vocabulary. All vocabulary names are linked by these fields to allow control for FORGETing through multiple vocabularys.

VOCABULARY 46

A defining word used in the form:

VOCABULARY cccc

to create a vocabulary definition cccc. Subsequent use of cccc will make it the CONTEXT vocabulary which is searched first by interpret. The sequence "cccc DEFINITIONS" will also make cccc the CURRENT vocabulary into which new definitions are placed.

In fig-FORTH, cccc will be so chained as to include all definitions of the vocabulary in which cccc is itself defined. All vocabularys ultimately chain to Forth. By convention, vocabulary names are to be declared IMMEDIATE. See VOC-LINK.

WARNING --- addr 20

A user variable containing a value controlling messages. If = 0, messages will be presented. If = -1, execute (ABORT) for a user specified procedure. See MESSAGE, ERROR.

WHILE f --- (run-time) 74

adl nl --- adk bk ad2 n2

Occurs in a colon-definition in the form:

BEGIN ... WHILE (tp) ... REPEAT

At run-time, WHILE selects conditional execution based on boolean flag f. If f is true (non-zero), WHILE continues execution of the true part through to REPEAT, which then branches back to BEGIN. If f is false (zero), execution skips to just after REPEAT, exiting the structure.

At compile-time, WHILE emplaces (OBRANCH) and leaves ad2 of the reserved offset. The stack values will be resolved by REPEAT.

WIDTH ---- addr 20

In fig-FORTH, a user variable containing the maximum number of letters saved in the compilation of a definitions' name. It must be 1 through 31, with a default value of 31. The name character count and its natural characters are saved, up to the value in WIDTH. The value may be changed at any time within the above limits.

WORD c --- 37

Read the next text characters from the input stream being interpreted, until a delimiter c is found, storing the packed character string beginning at the dictionary buffer HERE. WORD leaves the character count in the first byte, the characters, and ends with two or more blanks. Leading occurrances of c are ignored. If BLK is zero, text is taken from the terminal input buffer, otherwise from the disc block stored in BLK. See BLK, IN.

WRTFILE 66

Primative for moving a block from a block buffer to a system text file in RAM.

X 35

This is pseudonym for the "null" or dictionary entry for a name of one character of ascii null. It is the execution procedure to terminate interpretation of a line of text from the terminal or within a buffer, as both buffers always have a null at the end.

XOR nl n2 --- xor 12

Leave the bitwise logical exclusive-or of two values.

[Used in a colon-definition in form: 29

: xxx [words] more ;

Suspend compilation. The words after [are executed, not compiled. This allows calculation or compilation exceptions before resuming compilation with]. See LITERAL,].

[COMPILE] 43

Used in a colon-definition in form:

: xxx [COMPILE] FORTH ;

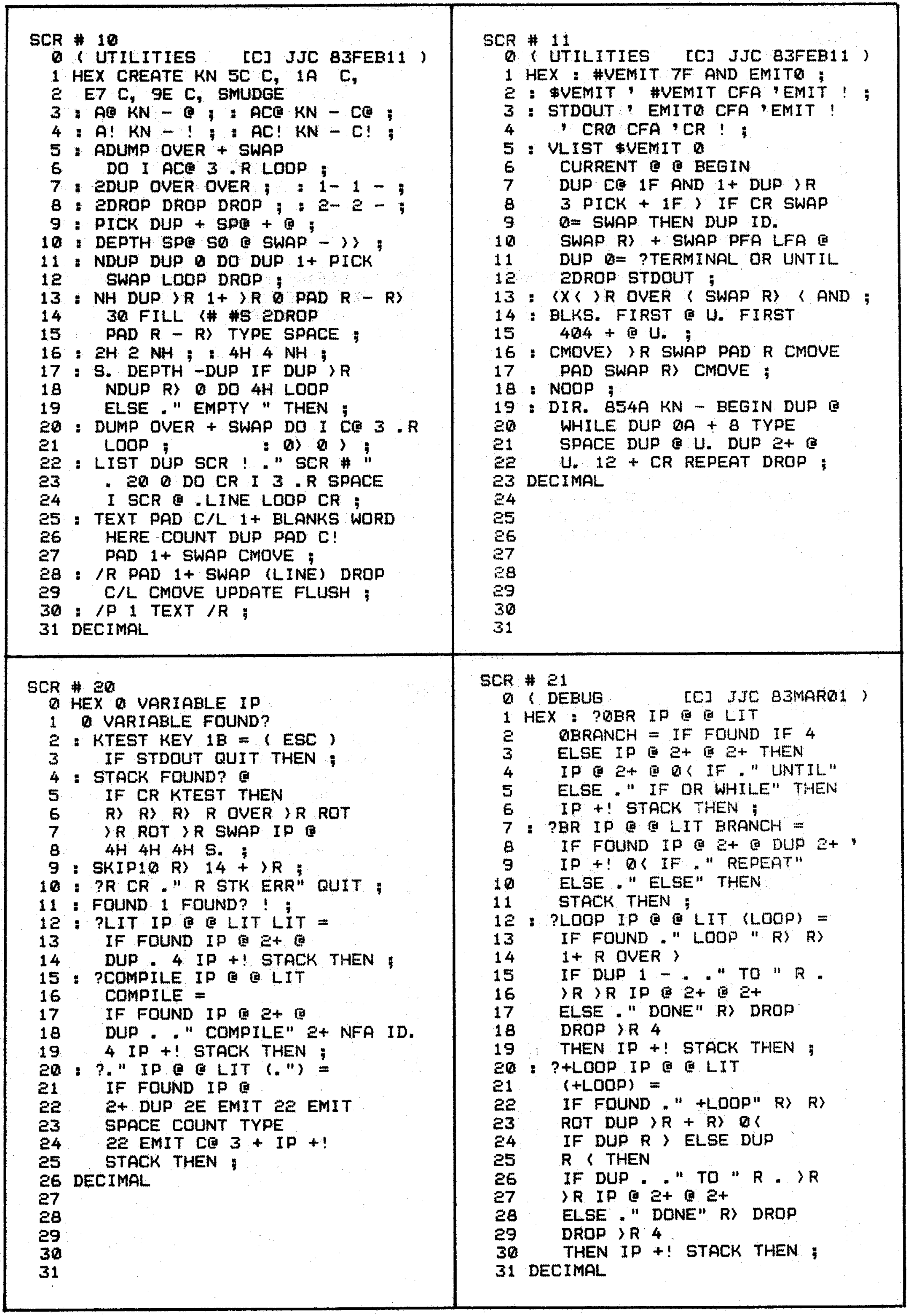
[COMPILE] will force the compilation of an immediate definition, that would otherwise execute during compilation. The above example will select the FORTH vocabulary when xxx executes, rather than at compile time.

[NUMBER] addr --- d 39

Convert a character string left at addr with a proceeding count, to signed double number, using the current numeric base. If a decimal point is encountered in the text, its position will be given in DPL, but no other effect occurs. If numeric conversion is not possible an error message will be given.

] 29

Resume compilation, to the completion of a colon-definition. See [.



```
SCR # 10
  0 ( UTILITIES     [C] JJC 83FEB11 )
  1 HEX CREATE KN 5C C, 1A  C,
  2  E7 C, 9E C, SMUDGE
  3 : A@ KN - @ ; : AC@ KN - C@ ;
  4 : A! KN - ! ; : AC! KN - C! ;
  5 : ADUMP OVER + SWAP
  6    DO I AC@ 3 .R LOOP ;
  7 : 2DUP OVER OVER ;  : 1- 1 - ;
  8 : 2DROP DROP DROP ; : 2- 2 - ;
  9 : PICK DUP + SP@ + @ ;
 10 : DEPTH SP@ S0 @ SWAP - >> ;
 11 : NDUP DUP 0 DO DUP 1+ PICK
 12    SWAP LOOP DROP ;
 13 : NH DUP >R 1+ >R 0 PAD R - R>
 14    30 FILL <# #S 2DROP
 15    PAD R - R> TYPE SPACE ;
 16 : 2H 2 NH ; : 4H 4 NH ;
 17 : S. DEPTH -DUP IF DUP >R
 18    NDUP R> 0 DO 4H LOOP
 19    ELSE ." EMPTY " THEN ;
 20 : DUMP OVER + SWAP DO I C@ 3 .R
 21    LOOP ;          : 0> 0 > ;
 22 : LIST DUP SCR ! ." SCR # "
 23    . 20 0 DO CR I 3 .R SPACE
 24    I SCR @ .LINE LOOP CR ;
 25 : TEXT PAD C/L 1+ BLANKS WORD
 26    HERE COUNT DUP PAD C!
 27    PAD 1+ SWAP CMOVE ;
 28 : /R PAD 1+ SWAP (LINE) DROP
 29    C/L CMOVE UPDATE FLUSH ;
 30 : /P 1 TEXT /R ;
 31 DECIMAL
```

```
SCR # 11
  0 ( UTILITIES    [C] JJC 83FEB11 )
  1 HEX : #VEMIT 7F AND EMIT0 ;
  2 : $VEMIT ' #VEMIT CFA 'EMIT ! ;
  3 : STDOUT ' EMIT0 CFA 'EMIT !
  4    ' CR0 CFA 'CR ! ;
  5 : VLIST $VEMIT 0
  6    CURRENT @ @ BEGIN
  7    DUP C@ 1F AND 1+ DUP >R
  8    3 PICK + 1F > IF CR SWAP
  9    0= SWAP THEN DUP ID.
 10    SWAP R> + SWAP PFA LFA @
 11    DUP 0= ?TERMINAL OR UNTIL
 12    2DROP STDOUT ;
 13 : <X< >R OVER < SWAP R> < AND ;
 14 : BLKS. FIRST @ U. FIRST
 15    404 + @ U. ;
 16 : CMOVE> >R SWAP PAD R CMOVE
 17    PAD SWAP R> CMOVE ;
 18 : NOOP ;
 19 : DIR. 854A KN - BEGIN DUP @
 20    WHILE DUP 0A + 8 TYPE
 21    SPACE DUP @ U. DUP 2+ @
 22    U. 12 + CR REPEAT DROP ;
 23 DECIMAL
 24
 25
 26
 27
 28
 29
 30
 31
```

```
SCR # 20
  0 HEX 0 VARIABLE IP
  1  0 VARIABLE FOUND?
  2 : KTEST KEY 1B = ( ESC )
  3    IF STDOUT QUIT THEN ;
  4 : STACK FOUND? @
  5    IF CR KTEST THEN
  6    R> R> R> R OVER >R ROT
  7    >R ROT >R SWAP IP @
  8    4H 4H 4H S. ;
  9 : SKIP10 R> 14 + >R ;
 10 : ?R CR ." R STK ERR" QUIT ;
 11 : FOUND 1 FOUND? ! ;
 12 : ?LIT IP @ @ LIT LIT =
 13    IF FOUND IP @ 2+ @
 14    DUP . 4 IP +! STACK THEN ;
 15 : ?COMPILE IP @ @ LIT
 16    COMPILE =
 17    IF FOUND IP @ 2+ @
 18    DUP . ." COMPILE" 2+ NFA ID.
 19    4 IP +! STACK THEN ;
 20 : ?." IP @ @ LIT (.") =
 21    IF FOUND IP @
 22    2+ DUP 2E EMIT 22 EMIT
 23    SPACE COUNT TYPE
 24    22 EMIT C@ 3 + IP +!
 25    STACK THEN ;
 26 DECIMAL
 27
 28
 29
 30
 31
```

```
SCR # 21
  0 ( DEBUG          [C] JJC 83MAR01 )
  1 HEX : ?0BR IP @ @ LIT
  2    0BRANCH = IF FOUND IF 4
  3    ELSE IP @ 2+ @ 2+ THEN
  4    IP @ 2+ @ 0< IF ." UNTIL"
  5    ELSE ." IF OR WHILE" THEN
  6    IP +! STACK THEN ;
  7 : ?BR IP @ @ LIT BRANCH =
  8    IF FOUND IP @ 2+ @ DUP 2+ '
  9    IP +! 0< IF ." REPEAT"
 10    ELSE ." ELSE" THEN
 11    STACK THEN ;
 12 : ?LOOP IP @ @ LIT (LOOP) =
 13    IF FOUND ." LOOP " R> R>
 14    1+ R OVER >
 15    IF DUP 1 - . ." TO " R .
 16    >R >R IP @ 2+ @ 2+
 17    ELSE ." DONE" R> DROP
 18    DROP >R 4
 19    THEN IP +! STACK THEN ;
 20 : ?+LOOP IP @ @ LIT
 21    (+LOOP) =
 22    IF FOUND ." +LOOP" R> R>
 23    ROT DUP >R + R> 0<
 24    IF DUP R > ELSE DUP
 25    R < THEN
 26    IF DUP . ." TO " R . >R
 27    >R IP @ 2+ @ 2+
 28    ELSE ." DONE" R> DROP
 29    DROP >R 4
 30    THEN IP +! STACK THEN ;
 31 DECIMAL
```

```
SCR # 22
( DEBUG         [C] JJC 83MAR01 )
HEX : STEP STACK BEGIN BEGIN
   0 FOUND? ! ?LIT
   ?COMPILE ?." ?0BR
   ?BR ?LOOP ?+LOOP FOUND?
   @ 0= UNTIL IP @ @ LIT
   ;S =  IF ." ; " 1
   ELSE IP @ @ DUP 2+ NFA
   ID. CR KTEST
   IP @ @ LIT [COMPILE] DOES> =
   IP @ @ LIT [COMPILE] (;CODE)
   = OR IF
   DROP [ HERE 12 + ] LITERAL
   >R IP @ >R ;S STACK 1
   ELSE EXECUTE SKIP10 ?R ?R
   ?R ?R ?R ?R ?R ?R ?R ?R
   2 IP +! STACK 0 THEN
   THEN UNTIL ;
: FIND- -FIND IF DROP CFA ELSE
   QUIT THEN ;
: DEBUG $VEMIT FIND- CR
   IP ! 0 FOUND? !
   ." IP    RTN       PARM WORD"
   CR STACK IP @ @ '
   QUIT CFA @ = IF 2 IP +!
   ." : " IP @ NFA ID. CR
   STEP THEN ; IMMEDIATE
DECIMAL
```

```
SCR # 30
( HP-75 ASM    [C] JJC 82OCT31 )
HEX VOCABULARY 75ASM IMMEDIATE
: ;CODE ?CSP COMPILE (;CODE)
   [COMPILE] [ [COMPILE] 75ASM ;
   IMMEDIATE
: CODE ?EXEC CREATE [COMPILE]
   75ASM !CSP ;
: C; CURRENT @ CONTEXT ! ?CSP
   SMUDGE ;
: LABEL 0 VARIABLE -2 ALLOT
   [COMPILE] 75ASM SMUDGE !CSP ;
       75ASM DEFINITIONS
 0 VARIABLE AX    0 VARIABLE DX
: X, HERE 2+  2 OVER C! OCTAL
   NUMBER DROP HEX ;
: A, X, C, ;          84 LATEST C!
: D, X, 40 OR C, ; 84 LATEST C!
: 1>2 DUP 100 / SWAP FF AND ;
: 1M <BUILDS C,
   DOES> C@ C, ;
: 2M <BUILDS C,
   DOES> C@ C, C, ;
: 3M <BUILDS C,
   DOES> C@ C, C, C, ;
: (NM) DX @ DUP DUP 20 <
   IF >> << 2+ ELSE 8 / 8 * 8 +
   THEN SWAP - 0 DO C, LOOP ;
: NM <BUILDS C,
   DOES> C@ C, (NM) ;
DECIMAL
```

```
SCR # 31
( HP-75 ASM    [C] JJC 82OCT31 )
HEX
   80 1M ELB         81 1M ELM
   82 1M ERB         83 1M ERM
   84 1M LLB         85 1M LLM
   86 1M LRB         87 1M LRM
   88 1M ICB         89 1M ICM
   8A 1M DCB         8B 1M DCM
 ( 8C 1M TCB         8D 1M TCM
   8E 1M NCB         8F 1M NCM )
   90 1M TSB         91 1M TSM
   92 1M CLB         93 1M CLM
   94 1M ORB         95 1M ORM
   96 1M XRB         97 1M XRM
   98 1M BIN         99 1M BCD
   9A 1M SAD       ( 9B 1M DCE
   9C 1M ICE         9D 1M CLE
 ) 9E 1M RTN         9F 1M PAD
   A0 1M LDB         A1 1M LDM
   A2 1M STB         A3 1M STM
   A4 1M LDBD        A5 1M LDMD
   A6 1M STBD        A7 1M STMD
   A8 2M LDB=        A9 NM LDM=
   AA 2M STB=        AB NM STM=
 ( AC 1M LDBI        AD 1M LDMI
   AE 1M STBI        AF 1M STMI
   B0 3M LDBD=       B1 3M LDMD=
   B2 3M STBD=       B3 3M STMD=
   B4 3M LDBDX       B5 3M LDMDX
   B6 3M STBDX       B7 3M STMDX
 ) DECIMAL
```

```
SCR # 32
( HP-75 ASM    [C] JJC 82OCT31 )
HEX
( B8 3M LDBI=       B9 3M LDMI=
  BA 3M STBI=       BB 3M STMI=
  BC 3M LDBIX       BD 3M LDMIX
  BE 3M STBIX       BF 3M STMIX )
  C0 1M CMB         B1 1M CMM
  C2 1M ADB         C3 1M ADM
  C4 1M SBB         C5 1M SBM
  C6 3M JSBX        C7 1M ANM
  C8 2M CMB=        C9 NM CMM=
  CA 2M ADB=        CB NM ADM=
  CC 2M SBB=        CD NM SBM=
  CE 3M JSB=        CF NM ANM=
( D0 3M CMBD=       D1 3M CMMD=
  D2 3M ADBD=       D3 3M ADMD=
  D4 3M SBBD=       D5 3M SBMD=
( D6  UNUSED )    ( D7 3M ANMD=
  D8 1M CMBD        D9 1M CMMD
  DA 1M ADBD        DB 1M ADMD
  DC 1M SBBD        DD 1M SBMD  )
( DE  UNUSED )    ( DF 1M ANMD )
  E0 1M POBD+       E1 1M POMD+
  E2 1M POBD-       E3 1M POMD-
  E4 1M PUBD+       E5 1M PUMD+
  E6 1M PUBD-       E7 1M PUMD-
( E8 1M POBI+       E9 1M POMI+
  EA 1M POBI-       EB 1M POMI-
  EC 1M PUBI+       ED 1M PUMI+
  EE 1M PUBI-       EF 1M PUMI-
) DECIMAL
```

```
SCR # 33
( HP-75 ASM   [C] JJC 82OCT31 )
HEX
 F6 CONSTANT Ø=
 F4 CONSTANT POS
 FA CONSTANT CS
: NOT 1+ ;
: THEN HERE OVER 1+ - SWAP C! ;
: IF C, HERE Ø C, ;
: ELSE FØ IF SWAP THEN ;
: BEGIN HERE ;
: UNTIL C, HERE 1+ - C, ;
: AGAIN FØ UNTIL ;
: WHILE IF ;
: REPEAT SWAP AGAIN THEN ;
FORTH DEFINITIONS
 1ABØ CONSTANT SAVFVM
 1ACØ CONSTANT GETFVM
DECIMAL
```

```
SCR # 4Ø
( FIX FILL    [C] JJC 83MAR25 )
HEX
CODE #FL D,24 A,32 POMD+
         D,22      POMD+
         D,2Ø      POMD+
              A,34 ADM
         D,22      ICM
BEGIN    D,22      DCM  Ø= NOT
WHILE    D,24 A,2Ø PUBD+
REPEAT             RTN C;
  ' #FL CFA PAD ! ' ;S CFA PAD 2
+ ! PAD ' FILL 4 CMOVE
( ' SWAP CFA PAD ! ' )R CFA PAD
2 + ! PAD ' FILL 4 CMOVE )
( USE 1ST SET OF CMDS TO FIX
  FILL, THE 2ND TO RESTORE FILL
   TO ORIG-- WARNING!! DO NOT
   FORGET BELOW #FL WHILE FIX
   IS INSTALLED )
DECIMAL
```

```
SCR # 41
( PRINT DRIVER[C] JJC 83MAR3Ø )
HEX 4575 CONSTANT PRNTCH
CODE PREMIT D,22 A,32 POMD+
      A,36 SAVFVM 1>2 JSBX
      D,22 A,32       STB
           PRNTCH 1>2 JSB=
      D,36 A,Ø6       POMD-
      A,36 GETFVM 1>2 JSBX
                      RTN C;
: PRCR ØD PREMIT ØA PREMIT ;
: PROUT ' PREMIT CFA 'EMIT !
   ' PRCR CFA 'CR ! ;
: PRLIST 2Ø ' C/L ! BASE @ SWAP
   DECIMAL PROUT LIST
   STDOUT BASE ! ;
DECIMAL
```

```
SCR # 5Ø
( LCD EDITOR  [C] JJC 83APRØ4 )
HEX FFFC CONSTANT LCD
 Ø VARIABLE COL
: LCDRDY BEGIN LCD AC@ 1 AND
   Ø= UNTIL ;
: LCDCLR 2ØØØ LCDRDY LCD A! ;
: LCDADR ABS 4F MIN 5Ø SWAP - ;
: PCUR LCDADR 8Ø OR
   LCDRDY LCD AC! ;
: OFFCUR 1 PCUR ;
: LCD! ABS 1F MIN LCDADR
   SWAP 1ØØ * + LCDRDY LCD A!
   LCDRDY OFFCUR ;
: PUTCUR COL @ PCUR ;
: PUTCHAR COL @ LCD! ;
: PUTLINE ( ADR) 2Ø Ø DO DUP I +
   C@ I LCD! LOOP DROP ;
DECIMAL
```

```
SCR # 51
( LCD EDITOR  [C] JJC 83APR03 )
HEX 0 VARIABLE ROW
: (CUR) ROW @ << << << << <<
   + ;
: CURSOR COL @ (CUR) ;
: BUF PREV @ 2+ ;
: >CUR< ROW @ SWAP CURSOR + 3FF
   AND 20 /MOD ROW ! COL !
   ROW @ = 0= IF BUF 0 (CUR) +
   PUTLINE THEN PUTCUR ;
: RTCUR 1 >CUR< ;
: LTCUR 3FF >CUR< ;
: DNCUR 20 >CUR< ;
: UPCUR 3E0 >CUR< ;
: HOME 400 CURSOR - >CUR< ;
: HRTAB 8 >CUR< ;
: HLTAB 3F8 >CUR< ;
: RETN 400 CURSOR 0 (CUR)
   - - >CUR< ;
: BACK* BL BUF CURSOR 1- + C!
   -1 COL +! BL
   PUTCHAR PUTCUR ;
: VDTAB 80 >CUR< ;
: VUTAB 380 >CUR< ;
DECIMAL
```

```
SCR # 52
( LCD EDITOR  [C] JJC 83MAR25 )
HEX
: >INS< 3FF OVER << - CURSOR >
   IF BUF CURSOR + >R R OVER
   OVER + 3FF CURSOR -
   4 PICK - CMOVE> R> SWAP
   BLANKS BUF 0 (CUR) +
   PUTLINE PUTCUR THEN ;
: INS 1 >INS< ;
: INS+ 4 >INS< ;
: INS++ 20 >INS< ;
: >DEL< 3FF OVER << -
   CURSOR > IF BUF CURSOR +
   OVER OVER + SWAP 3FF
   CURSOR - 4 PICK - CMOVE BUF
   400 + OVER - SWAP
   BLANKS BUF 0 (CUR) +
   PUTLINE PUTCUR THEN ;
: DEL 1 >DEL< ;
: DEL+ 4 >DEL< ;
: DEL++ 20 >DEL< ;
: CTEOL BUF CURSOR + 20 COL @
   - BLANKS 1B EMIT 4A EMIT ;
: DQUIT STDOUT QUIT ;
: UQUIT UPDATE FLUSH DQUIT ;
DECIMAL
```

```
SCR # 53
( SCR ED       [C] JJC 83MAR25 )
HEX 0 VARIABLE CMDTBL -2 ALLOT
( RT ARROW) 87 C, ' RTCUR CFA ,
( LT ARROW) 86 C, ' LTCUR CFA ,
( UP ARROW) 84 C, ' UPCUR CFA ,
( DN ARROW) 85 C, ' DNCUR CFA ,
( BACK    ) 08 C, ' BACK* CFA ,
( FET     ) 89 C, ' HOME  CFA ,
( / RT AR ) A7 C, ' HRTAB CFA ,
( / LT AR ) A6 C, ' HLTAB CFA ,
( RTN KEY ) 0D C, ' RETN  CFA ,
( / UP AR ) A4 C, ' VUTAB CFA ,
( / DN AR ) A5 C, ' VDTAB CFA ,
( I/R KEY ) 88 C, ' INS   CFA ,
( / I/R   ) A8 C, ' INS+  CFA ,
( ^ I/R   ) C8 C, ' INS++ CFA ,
( DEL KEY ) 8A C, ' DEL   CFA ,
( / DEL   ) AA C, ' DEL+  CFA ,
( ^ DEL   ) CA C, ' DEL++ CFA ,
( CLR KEY ) 8B C, ' CTEOL CFA ,
( / ^ TAB ) EE C, ' DQUIT CFA ,
( ESC KEY ) 1B C, ' UQUIT CFA ,
(         ) 00 C, ' NOOP  CFA ,
DECIMAL
( AR=ARROW;    /=SHIFT KEY;
   ^=CTL KEY  DN=DOWN
  RT=RIGHT    LT=LEFT )
```

```
SCR # 54
( LCD EDITOR  [C] JJC 83APR04 )
HEX : DOED >R R 1F 80 <X<
   IF R BUF CURSOR + C!
   R PUTCHAR RTCUR
   ELSE CMDTBL
   BEGIN DUP C@ DUP 0= SWAP
    R = >R R OR 0= R>
    IF OVER 1+ @ EXECUTE THEN
   WHILE 3 + REPEAT DROP
   THEN R> DROP ;
: (EDIT) 0 COL ! 0 ROW
   ! BUF CURSOR + PUTLINE PUTCUR
   BEGIN KEY DOED AGAIN ;
: EDIT BLOCK DROP (EDIT) ;
DECIMAL
```

```
SCR # 60
( TV DRIVER    [C] JJC 83APR03 )
HEX 419A CONSTANT KEYDSP
CODE TVEMIT D,22 A,32 POMD+
      A,36 SAVFVM 1>2 JSBX
      D,22 A,32       STB
           KEYDSP 1>2 JSB=
      D,36 A,06       POMD-
      A,36 GETFVM 1>2 JSBX
                      RTN C;
: TVCR 0D TVEMIT 0A TVEMIT ;
: TVOUT ' TVEMIT CFA 'EMIT !
   ' TVCR CFA 'CR ! ;
DECIMAL
```

```
SCR # 61
( SCR ED       [C] JJC 83MAR25 )
HEX 0 VARIABLE ROW
 0 VARIABLE COL
: CURSOR ROW @ << << << << <<
   COL @ + ;
0 VARIABLE BUF.OS
: BUF PREV @ 2+ BUF.OS @ + ;
: PUTCUR 1B EMIT 25 EMIT COL @
   EMIT ROW @ EMIT ;
: >CUR< CURSOR + 1FF AND 20 /MOD
   ROW ! COL ! PUTCUR ;
: RTCUR 1 >CUR< ;
: LTCUR 1FF >CUR< ;
: BACK* CURSOR @ IF 8 EMIT
   SPACE 8 EMIT LTCUR THEN ;
: DNCUR 20 >CUR< ;
: UPCUR 1E0 >CUR< ;
: HOME 0 COL ! 0 ROW ! PUTCUR ;
: HRTAB 8 >CUR< ;
: HLTAB 1F8 >CUR< ;
: RETN 0 COL ! 0D EMIT ;
: VDTAB 80 >CUR< ;
: VUTAB 180 >CUR< ;
: UPDN 1B EMIT 45 EMIT HOME BUF
   200 -TRAILING TYPE PUTCUR ;
: UPUP BUF.OS @ IF 0 BUF.OS !
   UPDN THEN ;
: DNDN BUF.OS @ 200 = 0= IF 200
   BUF.OS ! UPDN THEN ;
DECIMAL
```

```
SCR # 62
( SCR ED       [C] JJC 83MAR25 )
HEX
: >INS< 3FF OVER << - CURSOR >
   IF BUF CURSOR + >R R
   2DUP + 3FF CURSOR BUF.OS @
   + - >R R 4 PICK -
   CMOVE> 1B EMIT 4A EMIT
   R> R ROT BLANKS
   R> SWAP 1FF AND -TRAILING
   TYPE PUTCUR THEN ;
: INS 1 >INS< ;
: INS+ 4 >INS< ;
: INS++ 20 >INS< ;
: >DEL< 3FF OVER << -
   CURSOR > IF BUF CURSOR +
   >R R 2DUP + SWAP 3FF CURSOR
   BUF.OS @ + - >R R 4 PICK -
   CMOVE 1B EMIT 4A EMIT BUF
   400 + OVER - SWAP
   BLANKS R> R> SWAP 1FF AND
   -TRAILING TYPE PUTCUR THEN ;
: DEL 1 >DEL< ;
: DEL+ 4 >DEL< ;
: DEL++ 20 >DEL< ;
: CTEOL 20 COL @ - ROW @ F =
   IF 1- THEN BUF CURSOR +
   OVER BLANKS SPACES PUTCUR ;
: DQUIT 1 BUF.OS ! 0 COL ! 0E
   ROW ! PUTCUR STDOUT QUIT ;
: UQUIT UPDATE FLUSH DQUIT ;
DECIMAL
```

```
SCR # 63
( SCR ED       [C] JJC 83MAR25 )
HEX 0 VARIABLE CMDTBL -2 ALLOT
( RT ARROW) 87 C, ' RTCUR CFA ,
( LT ARROW) 86 C, ' LTCUR CFA ,
( UP ARROW) 84 C, ' UPCUR CFA ,
( DN ARROW) 85 C, ' DNCUR CFA ,
( BACK     ) 08 C, ' BACK* CFA ,
( FET      ) 89 C, ' HOME  CFA ,
( / RT AR ) A7 C, ' HRTAB CFA ,
( / LT AR ) A6 C, ' HLTAB CFA ,
( RTN KEY ) 0D C, ' RETN  CFA ,
( / UP AR ) A4 C, ' VUTAB CFA ,
( / DN AR ) A5 C, ' VDTAB CFA ,
( ^ UP AR ) C4 C, ' UPUP  CFA ,
( ^ DN AR ) C5 C, ' DNDN  CFA ,
( I/R KEY ) 88 C, ' INS   CFA ,
( / I/R    ) A8 C, ' INS+  CFA ,
( ^ I/R    ) C8 C, ' INS++ CFA ,
( DEL KEY ) 8A C, ' DEL   CFA ,
( / DEL    ) AA C, ' DEL+  CFA ,
( ^ DEL    ) CA C, ' DEL++ CFA ,
( CLR KEY ) 8B C, ' CTEOL CFA ,
( / ^ TAB ) EE C, ' DQUIT CFA ,
( ESC KEY ) 1B C, ' UQUIT CFA ,
(          ) 00 C, ' NOOP  CFA ,
DECIMAL
( AR=ARROW;    /=SHIFT KEY;
  ^=CTL KEY  DN=DOWN
  RT=RIGHT   LT=LEFT )
```

```
SCR # 64
  0 ( SCR ED        [C] JJC 83MAR25 )
  1 HEX
  2 : DOED >R 1F R < R 80 < AND IF
  3    CURSOR 1FE > IF HOME THEN
  4    R BUF CURSOR + C! R EMIT
  5    RTCUR ELSE CMDTBL
  6    BEGIN DUP C@ DUP 0= SWAP
  7     R = >R R OR 0= R>
  8     IF OVER 1+ @ EXECUTE THEN
  9    WHILE 3 + REPEAT DROP
 10    THEN R> DROP ;
 11 : (EDIT) TVOUT 1 BUF.OS ! UPUP
 12    BEGIN KEY DOED AGAIN ;
 13 : EDIT BLOCK DROP (EDIT) ;
 14 DECIMAL
 15
 16
 17
 18
 19
 20
 21
 22
 23
 24
 25
 26
 27
 28
 29
 30
 31
```

```
SCR # 70
  0 ( 75C DISASM  [C] JJC 82OCT31 )
  1 HEX
  2  BLK @ 2+ DUP CONSTANT NAMEBLK
  3  1+ CONSTANT BYTESBLK
  4  0 VARIABLE AX  0 VARIABLE DX
  5  0 VARIABLE LOC
  6 : REG DUP 40 < IF ." A," DUP AX
  7    ! ELSE ." D," 40 - DUP DX !
  8    THEN OCTAL 2H HEX 1 LOC +! ;
  9 : NAME 7F AND 8 * NAMEBLK BLOCK
 10    + 8 TYPE ;
 11 : 1BY NAME 1 LOC +! ;
 12 : 2BY LOC @ 1+ C@ SPACE 2H NAME
 13    2 LOC +! ;
 14 : 3BY LOC @ 2+ DUP C@ SPACE 2H
 15    1- C@ 2H NAME 3 LOC
 16    +! ;
 17 : NBY  DX @ DUP DUP 20 < IF <<
 18    >> 2+ ELSE 8 / 8 * 8 + THEN
 19    SWAP - >R LOC @ DUP R + SPACE
 20    DO I C@ 2H -1 +LOOP NAME
 21    R> 1+ LOC +! ;
 22 DECIMAL
 23
 24
 25
 26
 27
 28
 29
 30
 31
```

```
SCR # 71
  0 ( 75C DISASM  [C] JJC 82OCT31 )
  1 HEX
  2 : INSTR BEGIN LOC @ C@ DUP
  3    80 < WHILE REG REPEAT DUP 7F
  4    AND 10 /MOD 20 * SWAP 1+ +
  5    BYTESBLK BLOCK + C@ DUP
  6     31 = IF DROP 1BY ELSE DUP
  7     32 = IF DROP 2BY ELSE DUP
  8     33 = IF DROP 3BY ELSE
  9     4E = IF       NBY
 10          THEN THEN THEN THEN ;
 11 : ALINE LOC @ 4H SPACE INSTR ;
 12 : DISASM LOC ! BEGIN CR ALINE
 13    KEY 1B = IF QUIT THEN
 14    0 UNTIL ;
 15 DECIMAL
 16
 17
 18
 19
 20
 21
 22
 23
 24
 25
 26
 27
 28
 29
 30
 31
```

```
SCR # 72
  0 ELB     ELM     ERB     ERM
  1 LLB     LLM     LRB     LRM
  2 ICB     ICM     DCB     DCM
  3 TCB     TCM     NCB     NCM
  4 TSB     YSM     CLB     CLM
  5 ORB     ORM     XRB     XRM
  6 BIN     BCD     SAD     DCE
  7 ICE     CLE     RTN     PAD
  8 LDB     LDM     STB     STM
  9 LDBD    LDMD    STBD    STMD
 10 LDB=    LDM=    STB=    STM=
 11 LDBI    LDMI    STBI    STMI
 12 LDBD=   LDMD=   STBD=   STMD=
 13 LDBDX   LDMDX   STBDX   STMDX
 14 LDBI=   LDMI=   STBI=   STMI=
 15 LDBIX   LDMIX   STBIX   STMIX
 16 CMB     CMM     ADB     ADM
 17 SBB     SBM     JSBX    ANM
 18 CMB=    CMM=    ADB=    ADM=
 19 SBB=    SBM=    JSB=    ANM=
 20 CMBD=   CMMD=   ADBD=   ADMD=
 21 SBBD=   SBMD=   ???     ANMD=
 22 CMBD    CMMD    ADBD    ADMD
 23 SBBD    SBMS    ???     ANMD
 24 POBD+   POMD+   POBD-   POMD-
 25 PUBD+   PUMD+   PUBD-   PUMD-
 26 POBI+   POMI+   POBI-   POMI-
 27 PUBI+   PUMI+   PUBI-   PUMI-
 28 JMP     JNO     JOD     JEV
 29 JNG     JPS     JNZ     JZR
 30 JEN     JEZ     JNC     JCY
 31 JLZ     JLN     JRZ     JRN
```

```
SCR # 73
 1111111111111111
 1111111111111111
 111111112N2N1111
 3333333333333333
 1111113l2N2N2N3N
 333333?3111111?1
 1111111111111111
 2222222222222222
```

```
SCR # 80
( SIZE CHANGE [C] JJC 83MAR22 )
HEX 1F9F CONSTANT ALLOC
CODE #SZ D,22 A,32 POMD+
   A,36 SAVFVM 1>2 JSBX
         D,22 A,32 STM
   D,30
   LIMIT 10 - 1>2  LDM=
              A,34 ADM
         ALLOC 1>2 JSB=
         D,36 A,06 POMD-
   A,36 GETFVM 1>2 JSBX
                   RTN C;
: SIZE+ >R R #SZ
   R   73 ( OR+22)  +!
   R  3F3 ( SP!)    +!
   R 1073 ( ABORT)  +!
   R  F19 ( ?STK)   +!
   R  65C ( FIRST)  +!
   R  668 ( LIMIT)  +!
   R 1B7C ( USE)    +!
   R> 1B7E ( PREV)  +! ABORT ;
: MEM. DD88 854A KN - >R R
   @ R> 2+ @ + - . ;
DECIMAL
```

EXTENSIONS GLOSSARY

This glossary contains all word definitions in the source screens. The definitions are presented in the order of their ASCII sort. For further information see the introduction to the kernal glossary.

SCREEN

#FL addr qty byte --- Fill Fix 40
A machine language version of FILL that works with quantities of zero and one. Lines 10, 11 of screen 40 patch this version into FILL. Lines 12, 13 restore the Fig version which propagates patterns. This version is required for the screen editors and is prepatched in F10L16 and F10V16.

#SZ #bytes --- Chg Size 80
Primative for SIZE+. nbytes will be inserted at LIMIT-10. Uses system call ALLOC which does all the necessary file manipulation.

#VEMIT Byte --- ASCII UTIL 11
Strips sign bit, then executes EMITØ. Allows clean display and printing (use printer as a DISPLAY IS device) of VLIST. Patched into 'EMIT (the execution vector EMIT by $VEMIT.

$VEMIT --- UTIL 11
Execution vector switch. Switches EMIT to use #VEMIT.

(CUR) col# --- cursor-pos LCD Ed 51
Multiplies row number by 32 and adds column number. Primative for CURSOR.

(EDIT) ---- LCD Ed 54
---- SCR Ed 64
User word. Goes into the EDIT mode, editing the current screen (the one in PREV). Primative of EDIT.

(NM) --- 75 ASM 30
DOES> portion of NM. Inserts into the dictionary the correct number of bytes off the stack for multi-byte literal immediate instruction (LDM= , AMN= , etc.). Uses DX, the data register pointer, to figure the number of bytes required.

/P line # --- UTIL 10
Equivalent of P in Fig line editor. Puts the following text into line#.

/R line # --- UTIL 10
Equivalent of R in Fig line editor. Replaces line# from PAD.

0= --- b 75 ASM 33
Creates a conditional jump based on zero flag. Used only with IF, UNTIL, or WHILE. Used only in the assembler.

0> n --- f UTIL 10
Leaves a true flag if the number is greater than zero (positive), otherwise leaves a false flag.

1- n --- n-1 UTIL 10
Decrements the 16 bit number on the top of the stack by one.

1>2 n --- b b 75 ASM 30
Splits the top stack item into 2 stack levels. Used after label names in assembler instructions which take literals or immediate values; i.e. A;36 SAVFVM 1>2 JSBX on screen 80.

1BY --- DISASM 70
Processes the disassembly of a single byte instruction; i.e. BIN.

1M byte --- 75 ASM 30
A defining word which creates words which, when executed, create single byte machine code instructions. For example, 98 1M BIN creates a word call BIN with 98 in its parameter field. When BIN is executed in defining a CODE word like CODE TEST BIN RTN C; the BIN places 98 in the dictionary.

2- n --- n-2 UTIL 10
Decrements the 16 bit number on the top of the stack by two.

2BY --- DISASM 70
Processes the disassembly of a two byte instruction; i.e. LDB=

2DROP n1 n2 --- UTIL
Drops top two items from stack.

2DUP n1 n2 --- n1 n2 n1 n2 UTIL 10
Duplicates top two items on stack. Equivalent to OVER OVER.

2H n --- UTIL 10
Prints top stack number as a two digit number with leading zeros using current base. Used originally with ~~hex~~, ~~thus~~ the H, but will work with any base. See 4H.

2M byte --- 75 ASM 30
A defining word which creates words which, when executed, create two byte machine code instructions: the operator and a single byte operand. For example A8 2M LDB= creates a word called LDB= with A8 in its parameter field. When LDB= executes, for example CODE TEST D,22 41 LDB= RTN C; the LDB= places A8 41 in the dictionary.

3BY --- DISASM 70
Processes the disassembly of a three byte instruction; i.e. JSB=

3M b --- 75 ASM 30
A defining word which creates words which, when executed, creates three byte machine code instructions: the operator and two bytes of operand. For example CE 3M JSB= creates a word called JSB= with CE in its parameter field. When JSB= executes, for example CODE TEST 83F8 (KYIDLE) 1>2 JSB= RTN C; the JSB= places CE F8 83 in the dictionary.

4H n --- UTIL 10
Prints top stack number as a four digit number with leading zeros using the current base. See 2H

75 ASM --- 75 ASM 30
The name of the assembler vocabulary. Links to FORTH.

;CODE Used in the form: 75 ASM 30
cccc .... ;CODE
assembly mnemonics
Stop compilation and terminate a new defining word cccc by compiling (;CODE). Set the CONTEXT vocabulary to ASSEMBLER, assembling to machine code the following mnemonics.

When cccc later executes in the form:
cccc nnnn
the word nnnn will be created with its execution procedure given by the machine code following cccc. That is, when nnnn is executed, it does so by jumping to the code after nnnn. An existing defining word must exist in cccc prior to ;CODE.

<X< A B --- f UTIL 11
Tests for A<X<B. Leaves a true or false flag.

>CUR< b --- LCD Ed 51
b --- SCR Ed 61
Performs most of the cursor movements for the LCD and video editors. Controlled by cursor movement,. FET, RTN keys in editors.

>DEL< b --- LCD Ed 52
b --- SCR Ed 62
Deletes b characters of text following the cursor, sliding existing text to left to fill in. Controlled by DEL key in editors.

>INS< b --- LCD Ed 52
b SCR Ed 62
Inserts b blanks with screen text sliding existing text b bytes over to right. Text at the end of screen is lost. Controlled by I/R key in editors.

?+LOOP --- Debug 21
Primitive for DEBUG. Tests if IP points to (+LOOP); if so, processes the jump (unless its done) and prints line.

?." --- Debug 20
Primitive for DEBUG. Tests if IP points to (."); if so, processes the message and prints a line.

?ØBR --- Debug 21
Primitive for DEBUG. Tests if IP points to ØBRANCH; if so, processes the branch and prints a line.

?BR --- Debug 21
Primitive for DEBUG. Tests if IP points to BRANCH; if so, processes the branch and prints a line.

?COMPILE --- Debug 20
Primitive for DEBUG. Tests if IP points to COMPILE; if so, prints name of word compiled, skips it, and prints a line.

?LIT --- Debug 20
Primitive for DEBUG. Tests if IP points to LIT; if so, prints the value, skips over it and prints a line.

?LOOP --- Debug 21
Primitive for DEBUG. Tests if IP points to (LOOP); if so, processes the jump (unless its done) and prints a line.

?R --- Debug 20
Primitive for DEBUG. Prints stack error message.

A! n addr --- UTIL 10
(absolute address)
Store 16 bits of n at absolute address and absolute address +1

A@ addr --- n UTIL 10
(absolute address)
Leave 16 bit contents of absolute address and absolute address +1 on the stack.

A, --- 75 ASM 30
A compiling word used in the assembler to create instructions which load ARP. Two digits specifying the octal register name immediately follow A, i.e. A,02 or A,57. When executed A,57 converts the octal 57 to hex 2F and places it in the dictionary.

AC! b addr --- UTIL 10
(absolute address)
Store 8 bits of n at absolute address.

AC@ addr --- b UTIL 10
(absolute address)
Leave 8 bit contents of absolute address on the stack.

ADUMP addr qty --- UTIL 10
(absolute address)
Dumps qty bytes starting at absolute address addr.

AGAIN --- 75 ASM 33
A compiling word used in the assembler to emplace instruction for an unconditional relative jump JMP.

ALINE --- DISASM 71
Used by DISASM to process a single machine operation along with its operands, if any.

ALLOC --- n SIZE 80
A system subroutine which allocates additional bytes to a file. See SIZE+.

AX --- addr 75ASM 30
--- addr DISASM 70
Temporary storage for current value of ARP while assembling and disassembling.

BACK* --- LCD Ed 51
--- SCR Ed 61
Moves cursor back one position erasing the character.

BEGIN --- 75 ASM 33
A word used in the assembler to mark the start of a BEGIN ... UNTIL or BEGIN ... WHILE ... REPEAT sequence.

BLKS. --- UTIL 11
A convenient word to print out the block numbers of the screens currently in the block buffers.

BUF --- addr LCD ED 51
--- addr SCR Ed 61
Returns the address of the first byte of the current block buffer.

BUF.OS --- addr SCR Ed 61
A variable containing either 0 if the top 16 lines of the screen are being edited or a 200 hex if the bottom are.

BYTESBLK --- n DISASM 70
A constant representing the block number containing the array of bytes-per-instruction (screen #73).

C; --- 75 ASM 30
Used to terminate a CODE or LABEL definition. Analogous to ; restores context to current, unsmudges and checks stack for compiler security.

CMDTBL --- addr LCD Ed 53
--- addr SCR Ed 63
An array created using VARIABLE containing three byte entries. Byte one is the byte returned by KEY. Bytes two and three are the CFA of the word to be executed to service that key. This is an execution array, searched by DOED and delimited by 00. The final NOOP handles the case when control-space bar is pressed (generates a null).

CMOVE> from to qty --- UTIL 11
A non propagating CMOVE to high memory.

CODE --- 75 ASM 30
Defining word for machine language words. Used as CODE TEST ... C;

COL --- addr LCD Ed 50
--- addr SCR Ed 61
Variable containing current column location of cursor between 0 and 1F.

CS --- b 75 ASM 33
Creates a conditional relative jump based on carry flag. Used only with IF, UNTIL or WHILE. Used only in assembler. CS means carry set.

CTEOL --- LCD Ed 52
--- SCR Ed 62
Screen editor instruction which clears to end of line. Controlled by CLR key.

CURSOR --- n LCD Ed 51
--- n SCR Ed 61
Return the location of cursor relative to the beginning of screen (LCD Ed) or relative to the beginning of the display (SCR Ed).

D, --- 75 ASM 30
A compiling word used in the assembler to create instruction which loads DRP. Two digits specifying the octal register name immediately follow D, i.e. D,02 or D,57. When executed D,57 converts the octal 57 to hex 2F, or 's with 40 and places 6F in the dictionary.

DEBUG --- Debug 22
Word used to decompile colon-definitions showing the stack contents after each word which makes up the definition. Any parameters required on the stack by the word being debugged must precede the DEBUG command i.e. 6 DEBUG ALLOT. Debugging may be terminated before completion of the word by ESC (control BACK). The debugger will not handle user defined compiling words unless so modified. That is, user defined words like ." require a change to DEBUG to avoid crashing.

DEPTH --- n UTIL 10
Returns the number of levels currently on the stack. See S.

DEL --- LCD Ed 52
--- SCR Ed 62
Deletes one character, sliding all following text to the left. Activated by DEL key.

DEL+ Same as DEL except four characters are deleted. Activated by shift DEL keys.

DEL++ Same as DEL except 32 characters (one line) are deleted. Activated by control DEL keys.

DIR. --- UTIL 11
A convenience word to print a list of all files in the system directory.

DISASM addr --- DISASM 71
High level word to disassemble machine code. Base should be HEX to be meaningful.

DNDN --- SCR Ed 61
Vidio editor word which displays the lower half of the block buffer for editing. Invoked by control down-arrow.

DNCUR --- LCD Ed 51
--- SCR Ed 61
Shifts cursor down one line. Invoked by down-arrow.

DOED --- LCD Ed 54
--- SCR Ed 64
High level interpreter for editors. Gets a key. If its a character, displays it, otherwise searches CMDTBL. If it finds it, it executes it, otherwise ignores it. Does this until QUIT is executed.

DQUIT --- LCD Ed 52
--- SCR Ed 62
Word to exit editor without updating or flushing. See UQUIT. Invoked by shift control TAB.

DUMP addr qty --- UTIL 1-
Displays qty bytes using current BASE.

DX --- addr 75 ASM 30
--- addr DISASM 70
Temporary storage for current value of DRP while assembling and disassembling.

EDIT scr# --- LCD Ed 54
scr# --- SCR Ed 64
High level word to start editor.

ELSE --- 75 ASM 33
A word used in the assembler in a IF... ELSE ... THEN structure. Places a relative unconditional jump in the dictionary.

FIND- --- CFA Debug 22
A special version of -FIND used in DEBUG.

FOUND --- Debug 20
Sets FOUND? true.

FOUND? --- addr Debug 20
A variable used by STEP.

GETFVM --- n 75 ASM 33
A constant containing the address of a subroutine that restores the registers used by FORTH allowing safe execution of system routines. See SAVFVM

HLTAB --- LCD Ed 51
--- SCR Ed 61
Shifts cursor 8 positions to left. Invoked by shift left arrow.

HOME --- LCD Ed 51
--- SCR Ed 61
Editor command to return cursor to upper left corner of vidio screen (SCR Ed) or ROW Ø COL Ø (LCD Ed). Invoked by FET key.

HRTAB --- LCD Ed 51
--- SCR Ed 61
Editor command to move cursor 8 positions to right. Invoked by shift right arrow.

IF --- 75 ASM 33
An assembler conditional relative jump. Must be preceded by O= , SC or POS. Used in IF ... THEN or IF ... ELSE ... THEN structures.

INS --- LCD Ed 52
--- SCR Ed 63
An editor word which moves all text following the cursor one position to the right and inserts a space. The last character is lost. Activated by the I/R key.

INS+ Same as INS except it inserts 4 blanks and is activated by Shift I/R.

INS++ Same as INS except it inserts 32 blanks (one blank line) and is activated by control I/R keys.

INSTR --- DISASM 71
A low level disassembler word which fetches the op code, decodes it as to type using the byte number array and shifts control to the proper routine based on type.

IP --- addr Debug 20
A variable used by the debugger as a pseudo instruction pointer.

KEYDSP --- n 60
A constant containing the address of a system routine which activates the IL interface and sends a byte to the DISPLAY IS device.

KTEST --- Debug 20
Provides single stepping for debugger. Checks KEY for ESC which causes exit from debugger mode.

KN --- addr UTIL 10
A machine language routine that returns to the stack the current absolute address of the start of FORTH.

LABEL --- 75 ASM 30
A compiling word used by the assembler to create a subroutine reference. Must be terminated by C;

LCD --- n LCD Ed 50
A constant containing the absolute address of the LCD I/O port.

LCD! char pos --- LCD Ed 50
Puts character at position. Position is from 0 to 31 with 0 at the left end of LCD screen.

LCDADR pos --- pos LCD Ed 50
A low level word that converts FORTH LCD position to the calculator type expected by the hardware driver.

LCDCLR --- LCD Ed 50
An instruction that causes a fast clearing of the screen.

LCDRDY --- LCD Ed 50
A loop that waits for the LCD status to become ready so that a new character can be sent to the LCD.

LIST SCR# --- UTIL 10
A command which causes a screen to be listed on the LCD as 32 lines of 32 characters. Also used by PRLIST.

LOC --- addr DISASM 70
A variable which points to the next address to be disassembled.

LTCUR --- LCD Ed 51
--- SCR Ed 61
Editor command to move cursor one position to left. Invoked by left arrow key.

MEM. --- SIZE 80
Prints the amount of memory available. Should be the same as the operating system MEM command.

NAME --- DISASM 70
Looks up the mnemonic for the op code and prints it.

NAMEBLK --- DISASM 70
An array in a screen containing the mnemonics for all HP-75C op codes.

NBY --- DISASM 70
Processes the disassembly of multi-byte instructions where the number of bytes depends on DRP and can range from two to nine. For instance LDM=

NDUP [n levels] n --- [n level] [n level]
UTIL 10
Duplicates n level of the parameter stack. See S.

NH n1 n2 --- UTIL 10
A primitive for 2H and 4H. Prints n1 using n2 digits with leading zeros.

NM b --- 75 ASM 30
A defining word which creates words which, when executed,create multi-byte assembler instructions. i.e. an operator plus one to eight byte of operand. For example A9 NM LDM= creates a word called LDM= with A9 in its parameter field. When LDM= executes, for example CODE TEST D,40 20 20 20 48 54 52 4F 46 LDM= RTN C; the LDM= places A9 46 4F 52 54 48 20 20 20 in the dictionary (which will load R40 with "FORTH")

NOOP --- UTIL 11
An ocassionally useful do nothing word. Used in CMDTBL.

NOT b --- b 75 ASM 33
Modifies a condition (Ø= CS POS) for use by a conditional relative jump IF, WHILE, UNTIL.

OFFCUR --- LCD Ed 50
Turns off the cursor in the LCD display.

PCUR pos --- LCD Ed 50
Puts cursor at COL position pos.

PICK n --- UTIL 10
An n level OVER. Copies the nth stack level to the top of the stack.

POS --- b 75 ASM 33
Creates a conditional relative jump based on the flags indicating positive. Used only with IF, UNTIL, WHILE. Used only in assembler.

PRCR --- PR Driv 41
Send a carriage return, line feed sequence to the PRINTER IS device. Gets executed by CR.

PREMIT char --- PR Driv 41
Sends char to PRINTER IS device. Gets executed by EMIT

PRLIST scr# --- PR Driv 41
Lists screen on PRINTER IS device.

PRNTCH --- n PR Driv 41
A constant containing the absolute address of a system routine which sets up the IL interface and sends a character to the PRINTER IS device.

PROUT --- PR Driv 41
Execution vector switch. Changes EMIT to use PREMIT and CR to use PRCR.

PUTCHAR char --- 50
Displays char on the LCD display at current COL position.

PUTCUR --- LCD Ed 50
Displays the cursor at the current COL position in the LCD display.

PUTLINE addr --- LCD Ed 50
Displays 32 characters starting at addr in the LCD window.

REG b --- DISASM 70
Process an op code which loads ARP or DRP. Prints the instruction as A,xx or D,xx where xx is the octal register name.

REPEAT --- 75 ASM 33
Used by the assembler in a BEGIN ... WHILE ... REPEAT structure. Places an unconditional relative jump back to BEGIN.

RETN --- LCD Ed 51
--- SCR Ed 61
An editor command which causes the cursor to move to the left end of display. Invoked by RTN key.

ROW --- addr LCD Ed 51
--- addr SCR Ed 61
A variable containing the current ROW number where the cursor is positioned.

RTCUR --- LCD Ed 51
--- LCR Ed 61
Editor command to move cursor one position to the right. Activated by right arrow.

--- UTIL 10
Prints contents of stack, nondestructively.

SAVFVM --- n 75 ASM 33
Constant containing the address of a subroutine that saves the registers used by FORTH allowing safe execution of system routines. See GETFVM.

SIZE+ n --- SIZE 80
A word which increases the size of FORTH by n bytes, adjusting the necessary internal FORTH pointers and valves.

SKIP10 --- Debug 20
A protection feature of DEBUG.

STACK --- Debug 20
Prints IP, two levels of the return stack and the contents of the parameter stack.

STEP --- Debug 22
Causes DEBUG to step through a colon-definition, processing one instruction at a time, simulating operation of the high level word.

STDOUT --- UTIL 11
Execution vector switch. Switches EMIT and CR back to standard EMITØ and CRØ.

TEXT --- UTIL 10
From Fig-FORTH line editor. Accepts the following text to PAD.

THEN --- 75 ASM 33
Used by assembler to terminate IF ... THEN or IF ... ELSE ... THEN control structures.

TVCR --- TV Driver 60
Sends a carriage return, line feed sequence to the DISPLAY IS device. Is executed by CR.

TVEMIT char --- TV Driver 60
Sends char to DISPLAY IS device over the IL interface without driving the LCD display. Speeds up character display. Is executed by EMIT.

TVOUT --- TV Driver 60
Execution vector switch. Changes EMIT to use TVEMIT and CR to use TVCR.

UNTIL --- 75 ASM 33
An assembler conditional relative jump. Must be preceeded by 0= CS or POS. Used in BEGIN ... UNTIL control structures.

UPCUR --- LCD Ed 51
--- SCR Ed 61
Editor command to shift cursor up one line. Invoked by up arrow.

UPDN --- SCR Ed 61
Primitive command for implementation of UPUP and DNDN in Video Editor

UPUP --- SCR Ed 61
Video editor word which displays the upper half of the block buffer for editing. Invoked by control up arrow.

UQUIT --- LCD Ed 52
--- SCR Ed 62
Word to exit editor, updating and flushing. See DQUIT. Activated by ESC (shift BACK)

VDTAB --- LCD Ed 51
--- SCR Ed 61
Editor command to shift cursor down four lines. Activated by Shift down arrow.

VLIST --- UTIL 11
Lists the names of the definitions in the context vocabulary. Hitting any key will terminate this listing.

WHILE --- 75 ASM 33
An assembler conditional relative jump. Must be prceeded by Ø= CS POS. Used in IF ... THEN or IF ... ELSE ... THEN control structures.

X, --- 75 ASM 30
Primitive for implementating A, and D, which see.

FORTH AGE PRODUCTIONS
23 Altarinda Road, Suite 213
Orinda, CA 94563
(415) 254-5221