THE MICRO WORKS

COLORFORTH

P.O. BOX 1110 DEL MAR, CA 92014 714-942-2400

THE MICRO WORKS

C O L O R F O R T H

E R R A T A N O T I C E

There have been four changes made to the source listing in the accompanying Owner's Manual; these comprise Version 1.1 of Colorforth. The changes have been made to the ROM-PACK, and are only included here for your records.

Location	Version 1.1
$CBDD	$84
$DØFA	$31
$E59B	$62
$E5A9	$64

If you have any questions on this or the program itself, you may call Talbot Microsystems at (213) 376-9941.

P.O. BOX 1110 DEL MAR, CA 92014 714-942-2400

COLORFORTH
V 1.0
for RADIO SHACK COLOR COMPUTER

TALBOT MICROSYSTEMS
1927 Curtis Ave
Redondo Beach, CA 90278
(714) 781-0464

COLORFORTH

T. J. Zimmer and R. J. Talbot, Jr.

CONTENTS

1.0 INTRODUCTION AND STARTUP

This document describes the COLORFORTH system provided in a rom pack for the RADIO SHACK COLOR COMPUTER. It is an extension of the FORTH Interest Group (fig) model of the language FORTH.

1.1 REQUIREMENTS

SOFTWARE

Rom COLOR BASIC must be resident in the computer in order for COLORFORTH to function. EXTENDED BASIC is not required.

HARDWARE

The COLOR COMPUTER must have at least 4k bytes of memory. COLORFORTH will automatically adjust to more if you have it. To use the cassette interface, it is necessary that the computer have control over the motor.

1.2 DISCLAIMER

This software is sold as is. It is believed to be free of error, but as we all know, no large piece of software is ever completely free of problems, so BEWARE!

1.3 HOWEVER

If you find any errors in the software, please notify the distributer so that future versions may be corrected.

1.4 STARTUP

To start COLORFORTH:

1. Verify power is off.
   *** NEVER, NEVER INSERT OR REMOVE CARTRIDGES ***
   *** WITH THE POWER ON !!! ***
2. Plug the cartridge into the slot on the side.
3. ONLY THEN turn on the power.

The computer should start up with a message on the screen saying that COLORFORTH is in control and "BELL" tone should be heard if you have the sound turned up.

1.5 FIRST IMPRESSIONS

After COLORFORTH is started as described previously, the FORTH operating system is running, with its compiler, interpreter, and text editor. Several changes from BASIC may be noted. First, if the volume is turned up on your television, you will notice a 'tick' sound emitted as each key is pressed. This is to help reduce key errors by providing a positive feedback indication that a key has been sensed. Second, if you use the backwards arrow key to backspace the cursor and erase characters, you will notice that a "BELL" tone is emitted with each key stroke when the cursor is at the beginning of the line. If you do not wish to hear these tones, simply turn the volume of the television down.

1.6 WHAT IS COLORFORTH ?

COLORFORTH is an implementation of fig-FORTH with the addition of many FORTH-79 words plus many words which provide interfaces to features specific to the COLOR computer. In FORTH the term "word" refers to an identifiable function or command, which in some computer languages is referred to as a subroutine or procedure.

If you wish to run FORTH-79 software, you should carefully read the sections which describe how COLORFORTH differs from FORTH-79.

COLORFORTH contains many features which are supplements to fig-FORTH and FORTH-79. You have full control over the allocation of memory between dictionary space (for programs) and the virtual input/output buffers (where, e.g., text for source code is edited). This permits you to optimize the use of memory for your specific application.

Binary overlays (compiled programs) are supported. This permits you to save compiled programs on tape for later use. For a large application this can save the time to load the program from tape by about a factor of three to four.

Sound output is possible in two forms: single tones (as in BASIC) and a high speed procedure which permits arbitrary waveforms to be sent.

A full assembler is not included; however, the vocabulary and a few basic words are included which permit an advanced programmer to insert specially hand coded machine language routines. A full 6809 FORTH assembler will be available on cassette.

A FORTH DECOMPILER is included as a learning and debugging tool.

1.7 NEW USER OF FORTH OR A VETERAN ?

Are you a FORTH veteran? If so, skip the next paragraph.

Are you a newcommer to the FORTH environment? If so, you should purchase the book STARTING FORTH by L. Brodie. This is a good introduction to this new and facinating programming tool known as FORTH. It is published by Prentice-Hall and should be available from bookstores or your COLORFORTH distributer. COLORFORTH provides a complete programming environment which in many respects is far superior to BASIC. As you learn how to do FORTH programming, you will initially find some of the methods of doing things to be peculiar. The environment, however, has evolved over many years by expert programmers who honed the environment for efficiency and speed. Once you learn the fundamental methods of FORTH programming, you too will find that you can put your machine to work in sophisticated ways with far less programming effort than by using other computer languages. You may wish to skim through Chapter 2 to get an idea of how the COLORFORTH system is organized, but do not fret over the fact that some of it may sound obscure -- nothing in there is necessary for one to use the basic system. Come back to it later after you have gotten a feel for the way the system works.

If you are a FORTH veteran, then you may be interested in the system configuration which is described in Chapter 2.

1.8 WHAT IS THE FORTH INTEREST GROUP (FIG) ?

The FORTH INTEREST GROUP is an independent group of FORTH enthusiasts whose aims are to educate others and to promote FORTH. They may be contacted at

P O Box 1105
San Carlos, CA 94070 (415) 962-8653

They publish a newsletter FORTH DIMENSIONS ($12/yr) and have many other publications available.

2.0 COLORFORTH SYSTEM CONFIGURATION

The following describes some of the aspects of how the COLORFORTH system is organized and how it is similar and/or dissimilar to other FORTH systems.

NOTICE!! If you are a complete beginner to FORTH, you should probably skip this section for now, and go to the book STARTING FORTH which you should have purchased. Just start working through the book. Do not be afraid of experimenting! The program is in the ROM PACK and can not be destroyed by any programming error. Just press the RESET button at the right rear and restart if necessary.

ALSO!! Your ROM PACK contains fig-FORTH which is not identical to the version of the FORTH-79 which is discussed by the author of the book STARTING FORTH, so refer to the later section entitled FORTH-79 DIFFERENCES as you go through the book.

NOTATION --

The following symbol terminology is used throughout this manual.

SYMBOLS	MEANING
a1 or addr1	16 bit address
n, n1, n2	16 bit signed number
d1, d2	32 bit signed double number
u1, u2	16 bit unsigned number
b1	8 bit byte -- unsigned
c	7 bit ascii character
f	Boolean flag
tf	True boolean flag -- 1
ff	False boolean flag - 0
string1, t	Ascii text string

STACK VALUES (b -- t ; a)

b	the stack before the word executes
--	the word being defined
t	the string which follows the word in some cases
;	denotes the place where the <ENTER> key would be pressed
a	the stack after the word executes

STACK PARAMETER DESCRIPTIONS

n1\n2\n3

The value n1 was placed on the stack first, then n2, then n3. The notation is read "n1 under n2 under n3" . n1 is on top.

E.g.

In the description of a FORTH word called <name>, the following might appear:

a1\n1\n2 -- t ; a2

which is interpreted as follows: before the word <name> is executed there are three items on the stack with n2 being the last placed on, n1 the previous one, and a1 the earliest one. Following <name> in the keyboard input is the text string t followed by pressing the <ENTER> key. Following the execution of the word <name>, the stack is left with the one item a2 .

2.1 MEMORY ALLOCATION

COLORFORTH will run on any RADIO SHACK COLOR COMPUTER with 4k, 16k, or 32k bytes of user read/write memory (RAM). The system is normally initialized with 8 screen buffers (only 1 on a 4k machine); each buffer is 1028 bytes in length, and holds one FORTH SCREEN (or BLOCK) of 1024 (1k) bytes of data plus 4 bytes of control information. The terms SCREEN and BLOCK are used somewhat interchangably; however, SCREEN is more appropriate in the case of text (displayed to the television screen) and BLOCK is a more abstract term useful in cases where the data is either text, numerical data, or machine code. The term buffer refers to one or more of the sets of machine memory locations dedicated to holding the SCREENs and BLOCKs. The existance of 8 buffers means that a program consisting of 8k of text may be edited or manipulated in memory, without resort to moving text to and from tape.

The maximum number of BLOCKS in memory (preset to 8 -- or 1 in the case of a 4k machine) is held in a variable named BMAX . The FORTH word BLOCK will not permit you to request a block number greater than BMAX or less than 1. The value of BMAX may be changed by use of the word called #BLOCKS . This will cause the system to change the number of buffers it uses and the space reclaimed is used for compiled program dictionary space. E.g., to change the allocation to 3 buffers, use

```
3 #BLOCKS    <ENTER>
```

(The notation <ENTER> is used to denote the place in the input where the key on the terminal marked "ENTER" is pressed. Similarly, <BREAK> would indicate where the key marked "BREAK" is pressed.)

The above sequence will recover 5k bytes for additional dictionary space (until a RESET causes it to go back to default value). A similar operation could be done to increase the number of buffers used (and therefore, of course, reduce the amount of program storage available).

Some caution should be taken with the use of the #BLOCKS command. It changes not only the number of buffers, but it also moves the user ram pointer called DP . Any program you have compiled up to the time you use #BLOCKS will be destroyed.

The system stacks are located below the lowest screen buffer. The RETURN STACK shares a page (of 256 bytes) of memory with the TERMINAL INPUT BUFFER, so its size is limited to about one hundred levels of nesting. The DATA STACK is allocated about 2

pages, so it is limited to about two hundred data items.

2.2 BLOCK INPUT/OUTPUT

For those familiar with other FORTH implementations, the word BLOCK is the only fig-FORTH mass-storage word in the system. All lower words (those used by BLOCK in a typical fig-FORTH system) are omitted, since they are not needed for this cassette implementation. The function of all of those words is contained in the word (BLOCK) to which BLOCK points. The word BLOCK is placed into RAM upon startup, and it consists of a "hook"

```
: BLOCK (BLOCK) ;
```

This means that the definition of BLOCK may be changed. E.g., an enterprising programmer might create an interface to a disk unit and call the new disk word DSKBLK . Then the system BLOCK word may be patched to point to DSKBLK by

```
' DSKBLK CFA ' BLOCK !
```

2.3 BASIC ROM CALLS

FORTH word	calls	Function
EMIT	[$A002]	Send char to output dev.
KEY	$A1B1	Get char from keyboard
?TERMINAL	[$A000]	Test for key pressed
TONE	$A951	Generate tone
WRITE	$A65C	Open file
WRITE	$A290	Write byte to cassette
WRITE	$A2A8	Write rest to cassette
WRITE	$A444	Close the file
READ	$A629	Open the file
READ	$A186	Read a byte of data
CLS	$A91C	Clear screen
JSTK	$A9DE	Read joystick values

Note: the " [$XXXX] " notation used refers to an indirect call to the address pointed at by the contents of the location $XXXX.

2.4 CASSETTE INTERFACE

Five words have been included in the system to permit easy writing and reading of data to and from cassette. These are:

```
READ      ( n1      --      )
READS     ( n1\n2   --      )
WRITE     ( n1      --      )
WRITES    ( n1\n2   --      )
```

In these words, n1 is the first SCREEN or BLOCK to read or write. n2, when present, is the number of sCREENs or BLOCKs to read or write.

DO NOT FORGET TO SET THE RECORDER TO THE PLAY OR RECORD MODE, AS APPROPRIATE, BEFORE PRESSING THE <ENTER> KEY.

The fifth word is

```
CLOADS    ( n1      --      )
```

n1 screens will be loaded from cassette, and compiled or interpreted. Each SCREEN will be read into buffer 1 and then 1 LOAD is performed. Then the next screen is read and loaded, etc. This procedure was choosen so that only a single buffer would be required in order to have the system function with minimal memory. This means that a very large program may be compiled. The number n1 specifies the number of screens to be read and loaded, and it must match the actual number of screens on the cassette.

NOTE: For CLOADS to function properly, it must control the cassette motor. So, the motor control wires must be connected.

NOTE: In FORTH, spaces are extremely significant. In the use of the words above, e.g., 19 CLOADS , it is essential that there be a space between 19 and CLOADS .

2.5 DOWN-LOADED WORDS

Several words are downloaded from ROM into RAM so that parts of them may be changed. They occupy about 80 bytes and include:

```
(ABORT)     BLOCK
FORTH       ASSEMBLER       EDITOR
```

(ABORT) is moved to ram so that its run-time behavior may be changed by the user. BLOCK is moved to ram for the same reason. FORTH , EDITOR , and ASSEMBLER are VOCABULARIES and must be in RAM in order to function.

2.6 WHAT IS MISSING ?

There are several fig-FORTH words which are not in this system. For the most part these are words in the fig-FORTH vocabulary which are used for disk interfaces. They are not needed in this implementation. They are

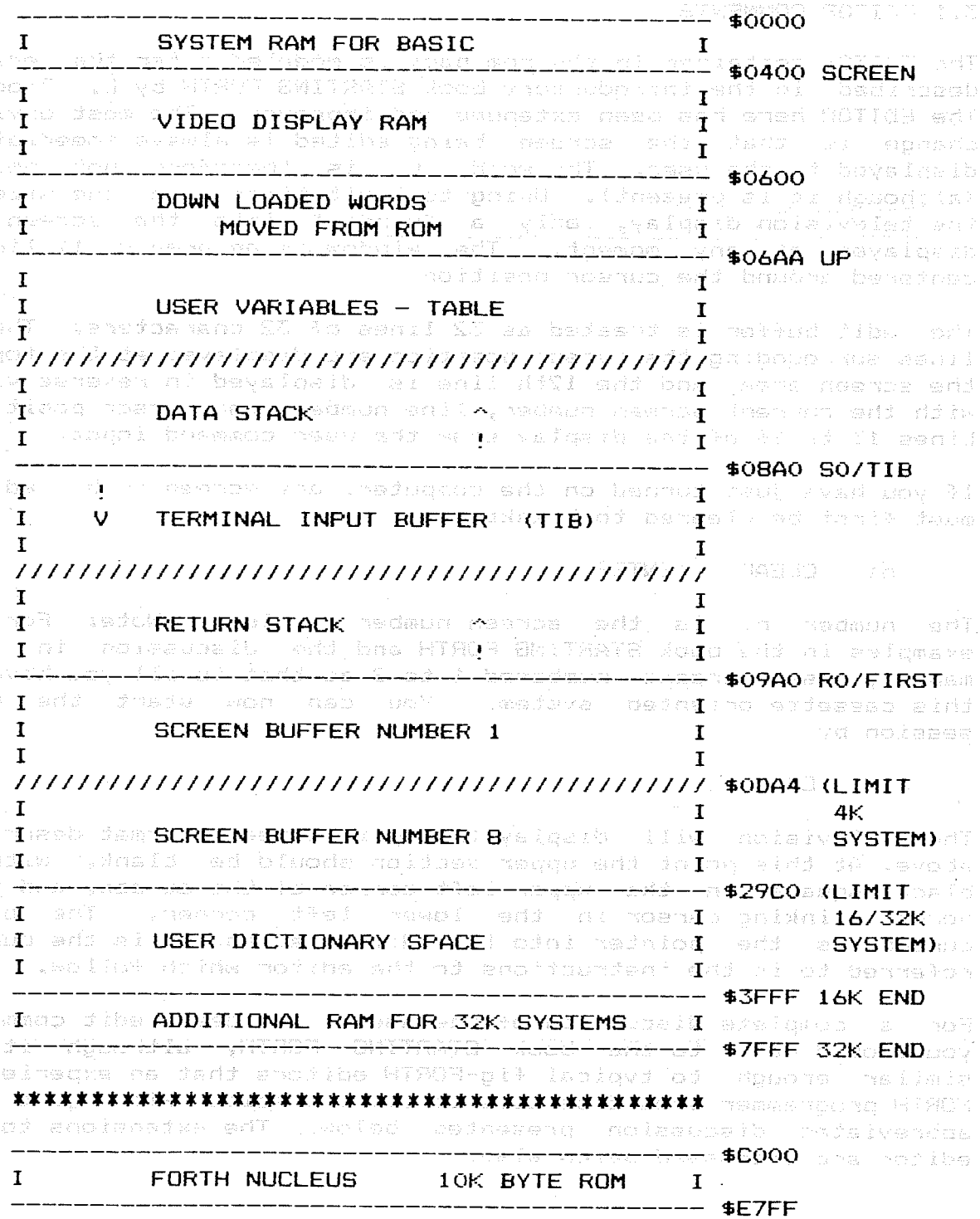
BUFFER +BUF R/W INDEX TRIAD PREV USE
FLUSH UPDATE EMPTY-BUFFERS

If desired, an experienced programmer can add these and change the hook for BLOCK as described above.

2.7 NEXT

For the interested assembly language programmer, NEXT is at $C052.

2.8 MEMORY MAP



3.0 EDITOR

3.1 EDITOR COMMENTS

The EDITOR contained in the rom pack is modeled after the editor described in the introductory book STARTING FORTH by L. Brodie. The EDITOR here has been extended and improved. The most obvious change is that the screen being edited is always immediately displayed to the user. The word L is therefore not needed (although it is present). Owing to limitations with the size of the television display, only a "window" into the screen is displayed at any moment. The window is an area of 11 lines, centered around the cursor position.

The edit buffer is treated as 32 lines of 32 characters. The 11 lines surrounding the cursor position are displayed at the top of the screen area, and the 12th line is displayed in reverse video with the current screen number, line number, and cursor position. Lines 13 to 16 of the display show the user command input.

If you have just turned on the computer, any screen to be edited must first be cleared to blanks by:

```
n1   CLEAR   <ENTER>
```

The number n1 is the screen number to clear. Note: For the examples in the book STARTING FORTH and the discussion in this manual, use screens numbered 1 to 8 as that is all you have in this cassette oriented system. You can now start the edit session by

```
n1 EDIT
```

The television will display the split screen format described above. At this point the upper section should be blank, with a black square in the upper left corner of the screen, and your normal blinking cursor in the lower left corner. The black cursor is the pointer into the edit area, and it is the cursor referred to in the instructions to the editor which follow.

For a complete discussion of the use of the basic edit commands you should refer to the book STARTING FORTH, although it is similar enough to typical fig-FORTH editors that an experienced FORTH programmer should be able to use the editor with just the abbreviated discussion presented below. The extensions to the editor are discussed below also.

3.2 LINE SPREADS

The two commands "SP" and "U" cause the text below the cursor to be moved down by one line. Since only a part of the edit buffer can be viewed at a time, there is a possibility that you will try to spread text off the bottom of the edit buffer. To prevent this, a protection word "?31" has been installed for use inside "SP" and "U" . This word causes a warning to be sent to the display if data will be lost from line 31, and then the system waits for your response. You may abort your command if you do not want to lose line 31.

3.3 EDITOR INTERNALS

This editor has its own interpreter, called QIT. This word allows the editor to split the screen into two parts and trap errors while remaining in the editor vocabulary. This is possible because the editor traps all errors by changing the function of (ABORT) to vector to ERR instead of ABORT. WARNING is changed to -1 to force all errors to go through (ABORT) to ERR. The word QUIT is redefined to restore WARNING and (ABORT) to their former state and then go to the FORTH QUIT . So, when in the EDITOR, to get back into FORTH you should type

```
QUIT <ENTER>
```

3.4 AN EDITOR EXAMPLE:

```
1 CLEAR
1 EDIT
3 T
P THIS IS A LINE OF TEEXT
WIPE
P THIS IS A LINE OF TEXT.
U HERE IS ANOTHER ONE
3 T
F EX
1 DEL
5 T
P THIS IS AANOTHER ONE ONE.
F AANOTHER
R ANOTHER LINE
D ONE
1 DEL
4 T
F A
TILL ER
8 T
P HERE IS A TEST LIST
F LIST
E
8 T
D TEST
I NEW LINE
3 T P EXAMPLE OF TWO COMMANDS
X
QUIT
```

3.5 EDITOR WORDS

WORD	(b -- t ; a)	FUNCTION
LINE EDITING COMMANDS:		
From STARTING FORTH editor:		
T	(n --)	Sets the cursor to start of line n.
P	(-- t;)	Text following space after P is placed into line pointed to by cursor.
U	(-- t;)	Text following space after U is inserted under the current line and all lower lines are moved down one.
M	(n1\n2 --)	Copies current line pointed to by cursor UNDER line n2 on block n1.
X	(--)	Deletes line pointed to by cursor and moves lower lines up one. Line 31 becomes blank.
Added from fig editors:		
H	(--)	Holds the line pointed at by cursor in PADI.
K	(--)	Kills (erases) the line pointed at by cursor.
SP	(--)	SPreads the lines --- the line pointed at by cursor and all following lines move down one. It tests line 31 for potential data loss. Current line becomes blank.
TOP	(--)	Moves cursor to TOP of edit screen.
BOT	(--)	Moves cursor to BOTtom of edit screen.
	(--)	Down arrow: Moves cursor to beginning of next lower line.
	(--)	Up arrow: Moves cursor to beginning of previous line.
	(--)	Double down arrow: Moves cursor to beginning of fourth (4th) line below current line.
	(--)	Double up arrow: Moves cursor to beginning of fourth (4th) line before current line.

STRING EDITING COMMANDS:
From STARTING FORTH editor:

F (-- t;) Finds first occurrence of the text following the blank after F. Starts at current cursor position.

S (n -- t;) Searchs for the text string following the command S. Starts on current screen and goes to screen n-1. Stops at each occurrence of the string and you then may either:
press <BREAK> to stop search and leave cursor pointing at string, or
press any other key to continue search.

E (--) Erases as many characters GOING BACKWARDS from cursor as are in the buffer PADF. Typically used after F or S.

D (-- t;) Deletes first occurrence of text following the command D , searching from cursor position to end of screen.

TILL (-- t;) Deletes all text starting at cursor position until and including the string following the command TILL . Works on current screen only. If string not found, no delete occurs.

I (-- t;) Inserts text following the command I into the edit buffer after the current position of the cursor. Text following is pushed off the end of line, and those at end are lost.

R (-- t;) Replaces the string just found (by, e.g., F) with the string following the command R .

Additions to editor:

+T (n --) Sets cursor to current line plus n.

C/L (-- n) Returns number of characters per line (32).

DEL (n --) DELetes n characters BEFORE the cursor and compresses the line to omit the space. Fills end of line with blanks.

C (n --) Cursor movement: moves forward (or backwards if n is negative) by n characters.

(R) (--) Text at PADI replaces the text in the current line.

(F) (--) Searches for the string in PADF, starting with the current cursor position till end of the screen.

(I) (--) Inserts the current contents of PADI into

edit buffer at the location of the cursor. Any text pushed off end of line is lost.

BMOV (a1 --) String at PAD is moved to address a1.

PADF (-- a1) Returns address of the Find buffer as a1.

PADI (-- a1) Returns address of the Insert buffer as a1.

PAD (-- a1) Returns address of scratchPAD area as a1.

TEXT (c -- t;) Accepts text following command TEXT into into scratchpad area PAD up to the character with ascii value c (or up to maximum of C/L characters or <ENTER>).

GTEXT (a1 --) Accepts text from input screen until a delimiting ^ is found or the <ENTER> key is pressed; text is moved to address a1.

FULL SCREEN ORIENTED COMMANDS:
From STARTING FORTH editor:

WIPE (--) Clears the current screen to all blanks.

COPY (n1\n2 --) Copies screen n1 to screen n2.

Added to the editor:

N (--) Go to Next higher edit screen and reset cursor to top of screen.

B (--) Go Back to previous screen and reset cursor to top of screen.

L (--) Lists the current screen. Not needed with crt screen oriented editor.

EDIT (n1 --) Sets EDITOR vocabulary, modifies (ABORT) to point to ERR, sets WARNING to -1, and then executes QIT.

CLEAR (n1 --) CLEARs screen n1 to all blanks.

NEW (--) Permits entry of NEW lines of text into current edit screen, staying in input mode till a null line is entered. EXAMPLE:

```
1 EDIT  <ENTER>        ( selects editor )
3 T <ENTER>            ( set cursor to line 3)
NEW <ENTER>            ( start NEW insert mode)
 THIS IS A TEST<ENTRY>
 THIS IS ANOTHER TEST<ENTRY>
 <ENTRY>                 ( a null line )
```

The two lines of text following NEW will be placed in lines 3 and 4 of screen \1. Previous contents of those lines are lost. The line with <ENTRY> only is a null line (i.e. it contains nothing) terminates the input mode; line \5 will be unchanged and the input mode will be exited back to the normal edit mode.

MISCELLANEOUS EDITOR WORDS

These are normally not ever directly used by a user, but an enterprising programmer who wishes to extend the EDITOR might use them.

-TEXT (a1\n1\a2 -- f) Primative string match routine. Matches string at a1 against string at a2, for a count of n1 characters. Flag f is returned true (1) if strings match; Otherwise, f is returned false (0). Written in assembly.

MATCH (a1\n1\a2\n2 -- tf\n3) for a match
-- ff\n1) for no match
String match routine which starts a search at a1 for a count of n1 characters. The string at a2 of length n2 is searched for as a substring of a1,n1 . If the string a2,n2 is found, a true flag (tf = 1) is returned along with n3 = the number of bytes from a1 to the end of the string match. If string a2,n2 is not found, a false flag (ff = 0) is returned, and n1 is on the stack.

1LINE (-- f) Uses MATCH to scan the current line for a match to the string in PADF . Returns flag f = true if found, false if not. Updates cursor position to end of matching string if found, or to end of line if no match.

-MOVE (a1\n1 --) MOVEs C/L characters from address a1 to line n1 of the current edit screen.

LINE (n1 -- a1) Returns the address a1 of the line n1 of the current edit screen.

#LOCATE (-- n1\n2) Returns the cursor location in the current edit screen. n2 is the line number, and n1 is the character position.

>L# (-- n1) Returns current line number as n1.

#LEAD (-- a1\n1) Returns the address of the cursor line as a1 and the position in the line as n1.

#LAG (-- a1\n2) Returns address of the cursor in the current screen as a1, and the number of characters from the current position to the line end as n2.

SERR (--) Issues message "NONE" for no string found.

QIT (--) EDITOR's INTERPRETER word. Scrolls the lower 4 lines only, views the current window into the edit buffer, and accepts and interprets commands from the operator. Is an infinite loop, terminated only by execution of QUIT .

ERR (n1 --) ERRor handler for the EDITOR; prints the error indication "<-?" and the error message and then goes to QIT.

QUIT (--) This is EDITOR's redefinition of the FORTH word QUIT . It is used to exit the EDITOR. It restores WARNING to 1, restores (ABORT) to point to ABORT, restores the condition of the display screen to its normal state, restores the vocabulary to FORTH, and then executes the FORTH QUIT .

SCROL (--) SCROLls the lower 4 lines of the display, and sets the cursor in the lower left corner of the screen.

RANGE (n1 -- n2) System word used to limit screen number n1 to the range 1 to BMAX; limited number is n2.

V (--) Calculates the window around the cursor and displays it on the screen. Window is the 11 lines surrounding the cursor. Then the cursor position in the current line is shown.

?31 (--) Test line 31 of current screen for being empty. If so, does nothing. If not empty, prompt operator with notice that data will be lost; go to QIT if user responds with "Y" so that no loss will occur.

!CUR (n1 --) Store n1 into variable #R (the cursor position) after limiting it to a range within the current edit screen (i.e., n1 is limited to 0 <= n1 <= 1023 .

4.0 fig-FORTH VOCABULARY

The following is a copy of the glossary of the words from the standard fig-FORTH Installation Manual. The notation used is similar to the notation defined earlier for use throughout the rest of this manual. There are words in this glossary which are not contained in COLORFORTH; they are indicated with an asterisk after their names. These are discussed previously in the section WHAT IS MISSING? They relate to disk input and output functions. COLORFORTH contains other words which are described in the EDITOR section or in the next sections.

This glossary contains all of the word definitions in Release 1 of fig-FORTH. 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 proceedure 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

The capital letters on the right show definition characteristics:

C	May only be used within a colon definition. A digit indicates number of memory addresses used, if other than one.
E	Intended for execution only.
LO	Level Zero definition of FORTH-78
L1	Level One definition of FORTH-78
P	Has precedence bit set. Will execute even when compiling.
U	A user variable.

Unless otherwise noted, all references to numbers are for 16 bit signed integers. On 8 bit data bus computers, the high byte of a number is on top of the stack, with the sign in the leftmost bit. For 32 bit signed double numbers, the most significant part (with the sign) is on top.

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

! n addr --- LO
Store 16 bits of n at address. Pronounced "store".

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

d1 --- d2 LO
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 LO
Terminates numeric output conversion by dropping d, leaving the text address and character count suitable for TYPE.

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

' --- addr P,LO
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".

(P,LO
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.

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

(;CODE) C
The run-time proceedure, 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 --- C2
The run-time proceedure compiled by +LOOP, which increments the loop index by n and tests for loop completion. See +LOOP.

(ABORT)
Executes after an error when WARNING is -1. This word normally executes ABORT, but may be altered (with care) to a user's alternative proceedure.

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

(FIND) addr1 addr2 --- pfa b tf (ok)
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 good match. If no match is found, only a boolean false is left.

(LINE) n1 n2 --- addr count
Convert the line number n1 and the screen n2 to the disc buffer address containing the data. A count of 64 indicates the full line text length.

(LOOP) C2
The run-time proceedure compiled by LOOP which increments the loop index and tests for loop completion. See LOOP.

(NUMBER) d1 addr1 --- d2 addr2
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 unconvertable digit. Used by NUMBER.

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

*/ n1 n2 n3 --- n4 LO
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 LO
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 LO
Leave the sum of n1+n2.

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

+- n1 n2 --- n3
Apply the sign of n2 to n1, which is left as n3.

+BUF * add1 --- addr2 f
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)
addr n2 --- (compile) P,C2,LO
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 to 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.

+ORIGIN n --- addr
Leave the memory address relative by n to the origin parameter area. n is the minimum address unit, either byte or word. This definition is used to access or modify the boot-up parameters at the origin area.

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

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

--> P,LO
Continue interpretation with the next disc screen. (pronounced next-screen).

-DUP n1 -- n1 (if zero)
n1 -- n1 n1 (non-zero) LO
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)
--- ff (not found)
Accepts the next text word (delimited by blanks) in the input strean 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
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 --- LO
Print a number from a signed 16 bit two's complement value, converted according to the numeric BASE. A trailing blanks follows. Pronounced "dot".

." P,LO
Used in the form:
." cccc"
Compiles an in-line string cccc (delimited by the trailing ") with an execution proceedure 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 may be an installation dependent value. See (.").

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

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

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

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

0 1 2 3 --- n

These small numbers are used so often that is is attractive to define them by name in the dictionary as constants.

0< n --- f LO

Leave a true flag if the number is less than zero (negative), otherwise leave a false flag.

0= n --- f LO

Leave a true flag is the number is equal to zero, otherwise leave a false flag.

OBRANCH f --- C2

The run-time proceedure 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 L1

Increment n1 by 1.

2+ n1 --- n2

Leave n1 incremented by 2.

: P,E,LO

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.

; P,C,LO

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

;CODE P,C,LO

Used in the form:

: cccc .... ;CODE
assembly mnemonics

Stop compilation and terminate a new defining word cccc by compiling (;CODE). Set the CONTEXT vocabulary to ASSEMBER, 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 proceedure given by 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.

;S P,LO

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 proceedure.

< n1 n2 --- f LO

Leave a true flag if n1 is less than n2; otherwise leave a false flag.

<# LO

Setup for pictured numeric output formatting using the words:

<# # #S SIGN #>

The conversion is done on a double number producing text at PAD.

<BUILDS C,LO

Used within a colon-definition:

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

Each time cccc is executed, <BUILDS defines a new word with a high-level execution proceedure. Executing cccc in the form:

cccc nnnn

uses <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 proceedures to written in high-level rather than in assembler code (as required by ;CODE).

= n1 n2 --- f LO

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

> n1 n2 --- f LO

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

>R n --- C,LO

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 -- LO

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

?COMP

Issue error message if not compiling.

?CSP

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

?ERROR f n ---
Issue an error message number n, if the boolean flag is true.

?EXEC
Issue an error message if not executing.

?LOADING
Issue an error message if not loading

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

?STACK
Issue an error message is the stack is out of bounds. This definition may be installation dependent.

?TERMINAL --- f
Perform a test of the terminal keyboard for actuation of the break key. A true flag indicates actuation. This definition is installation dependent.

@ addr --- n LO
Leave the 16 bit contents of address.

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

ABS n --- u LO
Leave the absolute value of n as u.

AGAIN addr n --- (compiling) P,C2,LO
Used in a colon-definion 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 --- LO
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 LO
Leave the bitwise logical and of n1 and n2 as n3.

B/BUF --- n
This constant leaves the number of bytes per disc buffer, the byte count read from disc by BLOCK.

B/SCR --- n
This constant leaves the number of blocks per editing screen. By convention, an editing screen is 1024 bytes organized as 16 lines of 64 characters each.

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

BASE --- addr U,LO
A user variable contaning the current number base used for input and output conversion.

BEGIN --- addr n (compiling) P,LO
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 correspoinding 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
A constant that leaves the ascii value for "blank".

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

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

BLOCK n --- addr LO
Leave the memory address of the block buffer containing block n. If the block is not already in memory, it is transferred from disc to which ever buffer was least recently written. 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

BLOCK-READ
BLOCK-WRITE These are the preferred names for the installation dependent code to read and write one block to the disc.

BRANCH C2,L0
The run-time proceedure 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
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 disc The block is not read from the disc. The address left is the first cell within the buffer for data storage.

C! b addr ---
Store 8 bits at address. On word addressing computers, further specification is necessary regarding byte addressing.

C, b ---
Store 8 bits of b into the next available dictionary byte, advancing the dictionary pointer. This is only available on byte addressing computers, and should be used with caution on byte addressing minicomputers.

C@ addr --- b
Leave the 8 bit contents of memory address. On word addressing computers, further specification is needed regarding byte addressing.

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

CMOVE from to count ---
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. Further specification is necessary on word addressing computers.

COLD
The cold start proceedure to adjust the dictionary pointer to the minimum standard and restart via ABORT. May be called from the terminal to remove application programs and restart.

COMPILE C2
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 additon to simply compling an execution address (which the interpreter already does).

CONSTANT n --- L0
A defining word used in the form:
n CONSTANT cccc
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 U,L0
A user variable containing a pointer to the vocabulary within which dictionary searches will first begin.

COUNT addr1 --- addr2 n L0
Leave the byte address addr2 and byte count n of a message text beginning at address addr1. It is presumed that the first byte at addr1 contains the text byte count and the actual text starts with the second byte. Typically COUNT is followed by TYPE.

CR L0
Transmit a carriage return and line feed to the selected output device.

CREATE
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 vocablary.

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

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

D+- d1 n --- d2
Apply the sign of n to the double number d1, leaving it as d2.

D. d --- L1
Print a signed double number from a 32 bit two's complement value. The high-order 16 bits are most accessable on the stack. Conversion is performed according to the current BASE. A blank follows. Pronounced D-dot.

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

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

DECIMAL L0
Set the numeric conversion BASE for decimal input-output.

DEFINITIONS L1
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)
c n1 --- ff (bad)
Converts the ascii character c (using base n1) to its binary equivalent n2, accompanied by a true flag. If the conversion is invalid, leaves only a false flag.

DLIST
List the names of the dictionary entries in the CONTEXT vocabulary.

DLITERAL d --- d (executing)
d --- (compiling) P
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
Convert d1 to its double number two's complement.

DO n1 n2 --- (execute)
addr n --- (compile) P,C2,LO
Occurs in a colon-definition in form:
DO ... LOOP
DO ... +LOOP
At run time, DO begins a sequence with repetitive execution controlled 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

urrent 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> L0
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 assembler, multi-diminsional arrays, and compiler generation.

DP ---- addr U,L
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 U,L0
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.

DR0
DR1 Installation dependent commands to select disc drives, by preseting OFFSET. The contents of OFFSET is added to the block number in BLOCK to allow for this selection. Offset is supressed for error text so that is may always originate from drive 0.

DROP n --- L0
Drop the number from the stack.

DUMP addr n --- L0
Print the contents of n memory locations beginning at addr. Both addresses and contents are shown in the current numeric base.

DUP n --- n n L0
Duplicate the value on the stack.

ELSE addr1 n1 --- addr2 n2
(compiling) P,C2,L0
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 --- L0
Transmit ascii character c to the selected output device. OUT is incremented for each character output.

EMPTY-BUFFERS * L0
Mark all block-buffers as empty, not necessarily affecting the contents. Updated blocks are not written to the disc. This is also an initialization proceedure before first use of the disc.

ENCLOSE addr1 c ---
ddr1 n1 n2 n3
The text scanning primitive used by WORD. From the text address addr1 and an ascii delimiting character c, is determined the byte offset to the 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 proceedure will not process past an ascii 'null', treating it as an unconditional delimiter.

END P,C2,L0
This is an 'alias' or duplicate definition for UNTIL.

ENDIF addr n --- (compile) P,C0,L0
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 ---
Clear a region of memory to zero from addr over n addresses.

ERROR line --- in blk
Execute error notification and restart of system. WARNING is first examined. If 1, the text of line n, relative to screen 4 of drive 0 is printed. This line number may be positive or negative, and beyond just screen 4. If WARNING=0, n is just printed as a message number (non disc installation). If WARNING is -1, the definition (ABORT) is executed, which executes the system ABORT. The user may cautiously modify this execution by altering (ABORT). 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 --
Execute the definition whose code field address is on the stack. The code field address is also called the compilation address.

EXPECT addr count --- L0
Transfer characters from the terminal to address, until a "return" or the count of characters have been received. One or more nulls are added at the end of the text.

FENCE --- addr U
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 ---
Fill memory at the address with the specified quantity of bytes b.

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

FLD --- addr U
A user variable for control of number output field width. Presently unused in fig-FORTH.

FORGET E,LO
Executed in the form:
FORGET cccc
Deletes definition named cccc from the dictionary with all entries physically following it. In fig-FORTH, an error message will occur if the CURRENT and CONTEXT vocabularies are not currently the same.

FORTH P,L1
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 LO
Leave the address of the next available dictionary location.

HEX LO
Set the numeric conversion base to sixteen (hexadecimal).

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

HOLD c --- LO
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 C,LO
Used within a DO-LOOP to copy the loop index to the stack. Other use is implementation dependent.
See R.

ID. addr ---
Print a definition's name from its name field address.

IF f --- (run-time)
--- addr n (compile) P,C2,LO
Occurs is 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 continues ahead thru the true part. If f is false (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
Mark the most resently 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 LO
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.

INDEX from to ---
Print the first line of each screen over the range from, to. This is used to view the comment lines of an area of text on disc screens.

INTERPRET
The outer text interpreter which sequentially executes or compiles text from the input stream (terminal or disc) 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 be given. Text input will be taken according to the convention for WORD. 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 LO
Leave the ascii value of the next terminal key struck.

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

LEAVE C,LO
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 prodeeds normally until LOOP or +LOOP is encountered.

LFA pfa --- lfa
Convert the parameter field address of a dictionary definition to its link field address.

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

LIST n --- LO
Display the ascii text of screen n on the selected output device. SCR contains the screen number during and after this process.

LIT --- n C2,LO
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) P,C2,LO
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 reusumed and LITERAL compiles this value.

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

LOOP addr n --- (compiling) P,C2,LO
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
A mixed magnitude math operation which leaves the double number signed product of two signed number.

M/ d n1 --- n2 n3
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 ud1 u2 --- u3 ud4
An unsigned mixed magnitude math operation which leaves a double quotient ud4 and remainder u3, from a double dividend ud1 and single divisor u2.

MAX n1 n2 --- max LO
Leave the greater of two numbers.

MESSAGE n ---
Print on the selected output device the text of line n relative to screen 4 of drive 0. n may be positive or negative. MESSAGE may be used to print incidental text such as report headers. If WARNING is zero, the message will simply be printed as a number (disc un-available).

MIN n1 n2 --- min LO
Leave the smaller of two numbers.

MINUS n1 --- n2 LO
Leave the two's complement of a number.

MOD n1 n2 --- mod LO
Leave the remainder of n1/n2, with the same sign as n1.

MON
Exit to the system monitor, leaving a re-entry to Forth, if possible.

MOVE addr1 addr2 n ---
Move the contents of n memory cells (16 bit contents) beginning at addr1 into n cells beginning at addr2. The contents of addr1 is moved first. This definition is appropriate on on word addressing computers.

NEXT
This is the inner interpreter that uses the interpretive pointer IP to execute compiled Forth definitions. It is not directly executed but is the return point for all code proceedures. It acts by fetching the address pointed by IP, storing this value in register W. It then jumps to the address pointed to by the address pointed to by W. W points to the code field of a definition which contains the address of the code which executes for that definition. This usage of indirect threaded code is a major contributor to the power, portability, and extensibility of Forth. Locations of IP and W are computer specific.

NFA pfa --- nfa
Convert the parameter field address of a definition to its name field.

NUMBER addr --- d
Convert a character string left at addr with a preceeding count, to a 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.

OFFSET --- addr U
A user variable which may contain a block offset to disc drives. The contents of OFFSET is added to the stack number by BLOCK. Messages by MESSAGE are independent of OFFSET. See BLOCK, DR0, DR1, MESSAGE.

OR n1 n2 -- or L0
Leave the bit-wise logical or of two 16 bit values.

OUT --- addr U
A user variable that contains a value incremented by EMIT. The user may alter and examine OUT to control display formating.

OVER n1 n2 --- n1 n2 n1 L0
Copy the second stack value, placing it as the new top.

PAD --- addr L0
Leave the address of the text output buffer, which is a fixed offset above HERE.

PFA nfa --- pfa
Convert the name field address of a compiled definition to its parameter field address.

POP
The code sequence to remove a stack value and return to NEXT. POP is not directly executable, but is a Forth re-entry point after machine code.

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

PUSH
This code sequence pushes machine registers to the computation stack and returns to NEXT. It is not directly executable, but is a Forth re-entry point after machine code.

PUT
This code sequence stores machine register contents over the topmost computation stack value and returns to NEXT. It is not directly executable, but is a Forth re-entry point after machine code.

QUERY
Input 80 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 L1
Clear the return stack, stop compilation, and return control to the operators terminal. No message is given.

R --- n
Copy the top of the return stack to the computation stack.

R# --- addr U
A user variable which may contain the location of an editing cursor, or other file related function.

R/W

addr blk f ---
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 determines the location on mass storage, performs the read-write and performs any error checking.

R>

--- n LO
Remove the top value from the return stack and leave it on the computation stack. See >R and R.

R0

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

REPEAT

addr n --- (compiling) P,C2
Used within a colon-definition in the form:
BEGIN ... WHILE ... REPEAT
At run-time, REPEAT forces an unconditional branch back to just after the correspoinding 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 LO
Rotate the top three values on the stack, bringing the third to the top.

RP!

A computer dependent proceedure to initialize the return stack pointer from user variable R0.

S->D

n --- d
Sign extend a single number to form a double number.

S0

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

SCR

--- addr U
A user variable containing the screen number most recently reference by LIST.

SIGN

n d --- d LO
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

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

SP!

A computer dependent proceedure to initialize the stack pointer from S0.

SP@

--- addr
A computer dependent proceedure 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

LO
Transmit an ascii blank to the output device.

SPACES

n --- LO
Transmit n ascii blanks to the output device.

STATE

--- addr LO,U
A user variable containg the compilation state. A non-zero value indicates compilation. The value itself may be implementation dependent.

SWAP

n1 n2 --- n2 n1 LO
Exchange the top two values on the stack.

TASK

A no-operation word which can mark the boundary between applications. By forgetting TASK and re-compiling, an application can be discarded in its entirety.

THEN

P,C0,LO
An alias for ENDIF.

TIB

--- addr U
A user variable containing the addr-ess of the terminal input buffer.

TOGGLE

addr b ---
Complement the contents of addr by the bit pattern b.

TRAVERSE

addr1 n --- addr2
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 hi memory; if n=-1, the motion is toward low memory. The addr2 resulting is address of the other end of the name.

TRIAD *
scr ---
Display on the selected output device the three screens which include that numbered scr, begining with a screen evenly divisible by three. Output is suitable for source text records, and includes a reference line at the bottom taken from line 15 of screen4.

TYPE
addr count --- LO
Transmit count characters from addr to the selected output device.

U*
u1 u2 --- ud
Leave the unsigned double number product of two unsigned numbers.

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

UNTIL
f --- (run-time)
addr n --- (compile) P,C2,LO
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 *
LO
Marks the most recently referenced block (pointed to by PREV) as altered. The block will subsequently be transferred automatically to disc should its buffer be required for storage of a different block.

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

USER
n --- LO
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
E,LO
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 U
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 FORGETting thru multiple vocabularys.

VOCABULARY
E,L
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 ulitmately chain to Forth. By convention, vocabulary names are to be declared IMMEDIATE. See VOC-LINK.

VLIST
List the names of the definitions in the context vocabulary. "Break" will terminate the listing.

WARNING
--- addr U
A user variable containing a value controlling messages. If = 1 disc is present, and screen 4 of drive 0 is the base location for messages. If = 0, no disc is present and messages will be presented by number. If = -1, execute (ABORT) for a user specified proceedure. See MESSAGE, ERROR.

WHILE
f --- (run-time)
adl nl --- adl nl ad2 n2 P,C2
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 contintues execution of the true part thru 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 U

In fig-FORTH, a user variable containing the maximum number of letters saved in the compilation of a definitions' name. It must be 1 thru 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 --- L0

Read the next text characters from the input stream being interpreted, until a delimiter c is found, storing the packed character string begining at the dictionary buffer HERE. WORD leaves the character count in the first byte, the characters, and ends with two or more blanks. Leading occurances 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.

X

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

XOR n1 n2 --- xor L1

Leave the bitwise logical exclusive-or of two values.

[P,L1

Used in a colon-definition in form:

```
: 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] P,C

Used in a colon-definition in form:

```
: xxx [COMPILE] FORTH ;
```

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

] L1

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

5.0 OVERLAYS

5.1 WHAT IS AN OVERLAY?

In COLORFORTH an overlay is a group of words which have been compiled into memory in a form which can be saved out to mass storage, in this case the cassette system.

5.2 WHY DO I NEED OVERLAYS?

As you become more familiar with COLORFORTH, you will write longer programs. For code more than a few screens long, the time to read and load the screens becomes noticably long. One can easily have programs with source code of dozens of screens, and these would take several minutes to read and load. What you gain by the use of overlays is the ability to save already compiled programs out on tape and the ability to load them back in without recompilation. Since a typical FORTH program is several times smaller in its compiled form, the compiled form occupies less tape which means it will take less time to read it in from the cassette. As an example, a 7k compiled overlay can be loaded in under two minutes while the source code for that would be over 40k bytes of text and would take over 14 minutes to read and load. QUITE A SAVINGS (of course you have to do it once!).

5.3 WHEN DO I MAKE AN OVERLAY?

It makes no sense to convert a program of 1 or 2 screens to overlay form. The time saved will not be significant, and for a 1 screen program no time will be saved because the compiled code is saved in units of 1k anyway.

You should make overlays only after you have a program fully debugged! Once compiled, you won't have the source to change. ALSO, BE SMART AND BE SURE YOU SAVE THE SOURCE CODE TO CHANGE LATER IF YOU NEED TO!!

If your program is over about 6 screens and it is working the way you want, then it is probably time to make an overlay.

5.4 HOW DO I MAKE AND SAVE AN OVERLAY?

Making an overlay is very easy. After you have written your program and have saved the source code on cassette in such a form that you can CLOADS it, you then type the following:

```
COLD <ENTER>
```

This will empty memory.

Now, you may want your program to run with a different number of screen buffers than 8, so you may optionally select a different number of buffers by performing:

n #BLOCKS <ENTER>

where n is the number of buffers you wish (minimum is 1). Now you can start the overlay process by typing:

OVSTART n1 CLOADS <ENTRY>

where n1 is the number of screens of source on the cassette. If you have previously gotten all the bugs out, the program should load ok. When the loading stops (you get the OK), terminate the overlay by typing

OVEND <ENTRY>

Actually, you may add to the program by typing it in before you type the OVEND. The overlay is now complete in memory, and so it is ready to be saved on cassette (A DIFFERENT CASSETTE!). Prepare a new tape for recording, and with the RECORD switch on type

OVSAVE <ENTRY>

This will save the compiled code out on tape. If you wish you can save it several times just to be safe.

**** NOTE ****

In our experience, it is not wise to try to store more than one set of text or one set of compiled overlay code on a single cassette tape. While it may seem wasteful of tape to put only one thing on a tape, it is EXTREMELY easy to mistakenly write over a portion of the tape that you did not wish to write on when you try to use a tape for more than one thing. For sure, you may use both sides of a tape, and you may place more than one copy of something on a tape as assurance that it is recorded properly at least once. YOU WILL make mistakes! So be liberal in your use of seperate tapes in order to minimize the loss of code when you do make them.

5.5 HOW DO I LOAD AN OVERLAY BACK IN?

Place the tape with the overlay into the cassette machine, rewind it, and place it in the PLAYBACK mode. Then type

COLD OVLOAD <ENTER>

COLD clears out memory, and OVLOAD reads the cassette. #BLOCKS will be reset to the value it had when the recording was made.

5.6 OVERLAY FACILITY GLOSSARY:

OVSTART (--) Signals start of an overlay. It creates an array of numbers in the beginning of memory to hold the parameters for the overlay. Some of the parameters held are:
BMAX DP LATEST
link to last name field in kernel
size of overlay.

OVEND (--) Completes the overlay setup in memory by filling in the incomplete array elements at the beginning of the overlay area.

OVLINK (--) Not used by the user in a direct manner. It is used by the system word OVLOAD to relink the dictionary after a new ovelay has been read in.

OVSAVE (--) Save to cassette tape as many BLOCKS as required to hold the overlay currently in memory between OVSTART and OVEND. The tape recorder must have been prepared to record. OVSAVE uses BLOCK #1 while saving, so its contents will be lost.

OVLOAD (--) Reads from cassette tape a full overlay and links it into memory. It sets #BLOCKS to the value it held when the overlay was created. It is suggested that COLD be executed before any overlay is loaded. An overlay may be of any length which will fit into memory.

?OV (--) Checks the first definition in memory following LIMIT to verify that an overlay is indeed present. An error message is generated if not.

6.0 SOUND WAVES

For those of you interested in creating complex sounds with the COLOR COMPUTER, COLORFORTH has two words called TONE and WAVE. The COLOR COMPUTER has within it a Digital-to-Analog Converter (DAC) with 6 bit resolution (64 values). The output of the DAC is connected to the sound input to the television display, so any output from it may be heard there. To try an example, turn up the volume so you can hear the clicking of the keys very clearly. Now type:

```
DECIMAL  49152  12  4000  WAVE  <ENTER>
```

The noise you hear is created by sending the first 4000 bytes of the COLORFORTH rom starting at memory location 49152 and holding each value with a duration value of 12 (which corresponds to a transfer rate of about 8 KHz).

The first number input to WAVE (49152 in this example) is the starting address of the table of values to send to the DAC, the second number is the duration of each hold value for the voltage (about 10 microseconds times the second number), and the last number is the number of bytes to send. The total duration of the sound is 10 microseconds times the product of the second and third numbers.

There is a table of values for a sine wave in the BASIC rom from address $A85C to $A87F. If this array is sent to WAVE, it will produce a single cycle of a sine wave. If one were to do that repeatedly, then a pure tone would be produced, and in fact TONE is a routine to do just that in an easier fashion. One could use that table as the basis of building up longer tables in memory. Some RADIO SHACK ROM games include VOICE! sent from a table in memory using this or a similar technique. It should be possible to receive voice through the JOYSTICK port by using a custom assembly language routine to build a voice array in memory, and then it may be sent back out by use of WAVE. HOW ABOUT ALL YOU INDUSTRIOUS EXPERIMENTORS OUT THERE!

6.1 SOUND SYSTEM GLOSSARY:

WAVE (a1\n2\n3 --) Custom wave form generation word. Used with a user prepared table of values. a1 is the start of a table of byte values, n2 is the time duration used, and n3 is the number of bytes in the table.

TONE (n1\n2 --) Generates a tone (through the television speaker) of frequency n1 and duration n2. This word uses the BASIC ROM routines. The values of n1 and n2 are the same as would be used in BASIC. TONE uses the system interrupts for timing, so it cannot be used with the time word ?TIME.

7.0 GRAPHICS

7.1 GRAPHICS CONTROL

The COLOR COMPUTER has many interesting graphics modes. There are several words in COLORFORTH to allow you to control and select these modes. See the glossary section on the GRAPHICS words for a list and definition of their functions.

The following is a list of the graphics modes available on the color computer and the values to go into the VCR and VDG registers to get them. For a detailed discussion, you should consult the manufacturer's specification sheet or see one of the articles published in various magazines on the details of the COLOR COMPUTER, e.g., the MARCH 81 BYTE magazine article.

MODE	VCR! (hex)	VDG!	RESOLUTION
Internal Alphanumeric	0	0	64 x 32
Semigraphics 6	2	0	64 x 48
Full Graphics 1 color	10	1	64 x 64
1 B/W	12	1	128 x 64
2 color	14	2	128 x 64
2 B/W	16	3	128 x 96
3 color	18	4	128 x 96
3 B/W	1A	5	128 x 192
6 color	1C	6	128 x 192
6 B/W	1E	6	256 x 192

Other modes are possible, but the above are the basic ones. The ones implemented in COLOR FORTH are the Full Graphics 2, 3, and 6 color modes. (The numbers 1, 2, 3, and 6 refer to the number of kilobytes of memory required for the display screen.)

To use the Full Graphics modes effectively, you will have to assign an array in memory to be the graphics screen and plot all of your graphics into that array. Here is a simple way to do that on 16k or larger machines:

```
DECIMAL   5 BLOCK 512 / 1+ 512 *  GSCR !  <ENTER>
```

This sets GSCR to point to an area which is 3k in size located between buffers 5 and 8, and which is forced to start on a 512 byte page boundry. To determine the page value for use in PG! , do the following:

```
GSCR @ 512 / CONSTANT NEW-PAGE
```

Now you may select the Full graphics 3 color mode and move the display to the new display area by executing the word VIEW defined by:

```
: VIEW  G3C  NEW-PAGE  PG !  ;
```

This will display the chosen graphics page, and if allowed to

continue, the system will immediately issue the OK back on the normal text display area. If you have just COLOR BASIC, the graphics screen will remain being displayed; however, if you have the EXTENDED BASIC ROM, then output to the normal text page causes the display to switch back there. Consequently, in the latter case, you must prevent the OK while you wish to view the graphics; this is conveniently done by placing KEY in the instructions where you wish the display to be held. Then, when you have finished observing the graphics, you can press any KEY to continue (it is good practice to DROP the character to keep the stack clean). So, to hold the displayed graphics screen, type

```
VIEW  KEY  DROP
```

Now, the graphics page will be displayed until you press any key. If you do not have the EXTENDED BASIC ROM, then you cause the display to switch back to the normal page with VNORM . A completely general form which will behave the same whether you have the EXTENDED BASIC ROM or not is

```
VIEW   KEY   DROP   VNORM
```

This will not be useful until you set up code to actually place something into the graphics page. To do that you must select a color and use the word SET to plot individual points. Rather than send the color for every point plotted, we set the color into the FORTH variable COLOR and then we must send only the x and y coordinates to SET. E.g., to use the RED color you would say:

```
RED COLOR !
```

First you will want to clear the graphics screen. In the case of using the G3C mode which uses 3k of memory (HEX 0C00 bytes), this would be done with this word CLR:

```
HEX  : CLR  GSCR @ 0C00  ERASE ;
```

Erasing the screen is equivalent to setting it to green.

As an example, lets now define a word which will draw a diagonal line and then look at it. Define this word DIAG

```
HEX  : DIAG CLR VIEW  40 0
         DO     I I  SET
         LOOP              KEY  DROP  VNORM ;
```

Now type the word DIAG and you should see a line drawn from the upper left (coordinate 0 0) to the lower right (coordiate hex 40 40). The display will stay until you press any key. The only function of the word KEY in the DIAG definition is to hold the display until you want to continue (and then the value of the key is DROPped). As soon as DIAG is finished, the system will send you OK and to do that it returns you to the normal screen.

THhe COLORFORT word SET will support only the color 2, 3, or 6 modes. The only thing which changes is the vertical resolution and the amount of memory required. To create another mode command, you may use the GMODE command as follows (e.g., to create G6B for 6k black and white):

```
HEX  1E  06  GMODE  G6B  <ENTER>
```

This command gives the highest possible resolution mode, but you will have to write your own version of SET to place the data in.

For the advanced programmer who wants to play with such techniques, the source code for the SET command is in screen 20 of the source code. It is in HEX code which was hand assembled into a CODE definition. If it were written in full FORTH ASSEMBLY language (an ASSEMBLER to do this will be available on cassette) it would be:

```
HEX   CODE  SET
            D  PULU,     20 # LDA,      MUL,      2 STD,
            D  PULU,   COLOR  LDA,     LSRB,
      CCLR  IF,                LSRA,    LSRA,
            ENDIF,             LSRB,
      CCLR  IF,                LSRA,    LSRA,
                               LSRA,    LSRA,
            ENDIF,           4 STA,     CLRA,
            2 ADDD,    GSCR  ADDD,    D X TRF,
            X LDA,        4  ORA,     X STA,    NEXT  C;
```

7.2 GRAPHICS SYSTEM GLOSSARY:

PG! (n1 --) Sets the 512 byte page number n1 into the system 6883 chip as the current video display page. CAUTION: The text output (e.g., via EMIT or TYPE) still goes to the same memory page as before. PG! is used to perform graphics.

VDG! (n1 --) Sets the graphics mode in the 6883 chip to the value n1. Only the lower three bits of n1 are used. See the chip's specification sheet or March 81 BYTE paper for programming this mode.

VCR! (n1 --) Sets the graphics mode of the 6847 VDG chip to the value n1. Only the lower 5 bits of n1 are used for this function. See the March 81 Byte article on the color computer for information on this mode selection.

VNORM (--) This restores the video display to the normal video mode. It is equivalent to

```
2 PG!   0 VDG!   0 VCR!
```

COLOR (-- a1) A system variable used by SET; contains the code for the COLOR to be plotted.

GREEN
YELLOW
BLUE
RED (-- n1) Constants which place on the stack the value n1 which is to be stored into color. E.g.,

```
RED  COLOR  !
```

will set the COLOR to be plotted to be RED.

GSCR (-- a1) A graphics system variable which holds the address of the first byte in the GRAPHICS SCREEN. This variable must be set before the SET command is executed. See the section on GRAPHICS CONTROL for further discussion.

GMODE (n1\n2 --) A Defining word which is used to create these graphics mode control words:

```
G2C   G3C   G6C
```

Users would normally not use GMODE unless they are advanced graphics programmers making use of the modes not otherwise supported by COLORFORTH .

G2C (--) Sets the GRAPHICS 2K BYTE COLOR mode. Sets the VDG and VCR to give a display of 128 x 64.

G3C (--) Sets GRAPHICS mode to 128 x 96.

G6C (--) Sets GRAPHICS mode to 128 x 192.

SET (x\y --) Set a point in the GRAPHICS SCREEN which begins at the location in GSCR. MUST!! have previously set GSCR in order for this word to work. The color of the point set is the color held in the variable COLOR, which must have been set also. The SET word supports the three previously specified graphics modes. If you define a new mode by using GMODE, the action of SET is undefined!

2nd argument of SET will come first

30 30 set Set point at 30,30

30 30 set Set point at 30,30

8.0 DEBUGGING AIDS

8.1 FORTH DECOMPILER

A very useful utility included in COLORFORTH is a decompiler. It allows you to take apart running FORTH code, including COLORFORTH itself! Of course, you have a listing of the source code for the whole system, but you will find the decompiler useful, especially when you forget the definition of some word.

Before explaining what the decompiler does, try it on your computer. Type in

    SOURCE ERASE <ENTER>

You should see the computer type out

    : ERASE 0 FILL ;S

That is the source code definition of ERASE. It fills a section of code with zeros. You will observe one thing which comprises one of the inner secrets of how FORTH operates -- the ; at the end of a colon definition inserts the code for ;S into the dictionary.

Try

    SOURCE HEX <ENTER>

You will get

    : HEX LIT 0 16 BASE ! ;S

In this case the definition has two bytes (0 and 16) which it does not recognize as a legitimate FORTH word, so it prints them out individually. Together they make up the 2 byte literal constant 16. The definition for HEX is

    : HEX 16 BASE ! ;

The FORTH compiler recognizes that 16 is not a FORTH word, and it inserts the word LIT together with the number 16 into the dictionary.

At this point you might wonder why 0 appeared in the SOURCE of ERASE; why wasn't it LIT 0 0 ? The answer is that 0 is a FORTH WORD !

The decompiler looks up each part of the word being decompiled and tries to locate that part in the dictionary. When it finds it, the name is printed using .NAME . You can control whether or not the addresses themselves are printed by using

    AON to turn on address printing

or AOFF to turn off address printing.

Try AON SOURCE HEX <ENTRY>.

The word SOURCE will trace only high level FORTH words, the ones made with : definitions. If you trace a CODE word like + you will just get a list of numbers --- it is a very slow way to get a memory dump! Just hold down any key to stop it.

The word ASOURCE is used by SOURCE. To use ASOURCE, you give it a parameter field address to start at E.g.

HEX CB16 ASOURCE <ENTER>

will also give you a trace of HEX, because CB16 is the parameter field address of HEX.

The sequence SOURCE XXX is exactly equivalent to

```
' XXX ASOURCE
```

8.2 DEBUGGING SYSTEM GLOSSARY:

DUMP (a1\n2 --) Dumps the contents of memory from a1 for a count of n2 characters using the current base. Prints addresses to left, then 8 bytes per line.

.NAME (a1\a2 -- a3) If a2 is the Code Field Address of some word, the name of the word is printed, and a3 = a1. Otherwise, it prints the high byte of a2 and sets a3 = a1-1.

.S (--) Nondestructive print of contents of stack.

AON (--) Turns on the printing of addresses by SOURCE.

AOFF (--) Turns off the printing of addresses by SOURCE.

SOURCE (-- t;) Decompiles the first word in the text string t .

ASOURCE (a1 --) Decompiles the FORTH word whose PFA is a1.

AFLG (-- a1) A system variable used to specify whether or not the address of a word will be printed during the decompiling action of SOURCE.

9.0 GLOSSARY OF OTHER WORDS IN COLORFORTH BUT NOT IN fig-FORTH

9.1 CASSETTE WORDS:

READ (n1 --) Reads the first block encountered on cassette starting at the present position of the tape. It is read into memory buffer number n1. BLOCKs numbers are not stored on cassette with the block data, so they can be read into any available screen buffer. E.g.

```
3 READ <ENTER>
```

This will read the first block encountered on the cassette into screen buffer number 3. The data is now available to FORTH by using

```
3 BLOCK
```

READS (n1\n2 --) Similar to READ, buts reads n2 blocks into memory buffers n1 ... n1+n2-1. E.g.

```
2 3 READS <ENTER>
```

will read 3 screens of data into buffers 2, 3, and 4.

WRITE (n1 --) Block n1 is written out to cassette at the position where the tape is set. You must correctly position the tape and set the recorder to the record mode before executing the command. E.g.

```
2 WRITE <ENTER>
```

WRITES (n1\n2 --) Writes blocks number n1 ... n1+n2-1 to the cassette. See READS .

CLOADS (n1 --) Reads n1 blocks from cassette sequentially, each going into block buffer #1, and after each is read, 1 LOAD is performed.

MOTOR (--) Similar to MOTOR in BASIC, but it simply toggles the motor to the opposite state each time it is executed; i.e., explicit ON or OFF is not stated.

9.2 LOADING WORDS:

THRU (n1\n2 --) BLOCKS n1 through n2 are LOADed in sequence.

TRY (--) The BLOCK specified in the variable SCR is LOADed. Since SCR is set when a screen is edited, after editing a screen one may just say

QUIT TRY

to test it.

9.3 MEMORY MANAGEMENT WORDS:

#BLOCK (n1--) Sets the number of BLOCK BUFFERS to n1 and sets DP to the end of the new buffer limit. It will not permit you to specify the number to be greater than the maximum allowed by the available memory:

size of memory	4k	16k	32k
number of buffers	1	14	29

EMPTY (--) Cleans out all words in the dictionary and and resets the DP variable to LIMIT.

BMAX (-- a1) Places on the stack the address a1 which contains the maximum valid BLOCK number which the system will allow. Its value may be changed to inform the system that it should reallocate the available memory to give a different number of BLOCK buffers.

FREE (-- n1) Returns n1, the number of free bytes of memory remaining. Computed by subtracting HERE from the contents of location $74 (which is set by the BASIC ROM upon reset). You can change the contents of location $74 yourself also.

9.4 STACK MANIPULATION WORDS:

S0 (-- a1) Returns a1, the address of a system variable which contains the address of the top of the data stack.

2OVER (d1\d2 -- d1\d2\d1) The second double number d1 is duplicated on top of the double number d2.

2SWAP (d1\d2 -- d2\d1) The two top double numbers are

reversed in order.

2DROP (d1 --) The top double number (two single numbers) are dropped from the stack.

2DUP (d1 -- d1\d1) The top double number is duplicated so that a second copy is on the stack also.

2* (n1 -- n2) The top number is doubled (does a left shift by one bit). Written in assembly.

2/ (n1 -- n2) The top number n1 is halved (does a right shift by one bit). Written in assembly.

'S (-- a1) Leaves the current address of the top of the stack on the stack (thus increasing depth of stack by one). Pseudonym for SP@ .

9.5 DISPLAY CONTROL WORDS:

? (a1 --) Prints the contents of the address a1 to the current output device using current base.

CUR (-- a1) A system variable which returns the address of location in memory which holds the cursor pointer. The sequence

```
CUR @
```

will leave the address of the current cursor position in the display on the stack.

CLS (n1 --) n1 is passed to the BASIC ROM routine which does a screen clear. n1 specifies the color the screen is to be set.

U. (n1 --) Prints number n1 as a 16 bit unsigned number.

H. (n1 --) Prints number n1 as a 16 bit unsigned hexadecimal number; useful for addresses in memory.

SCREEN (-- a1) Returns a1, the address of the upper left corner of the normal display screen ($400).

CHAN (-- a1) Leaves as a1 the address $6F which contains the channel variable used by the BASIC rom out put routines. Setting it to -2 will cause all output to go to the serial printer port; a value of 0 is normal crt output.

LISTS (n1\n2 --) Lists the screen n1 through n1+n2-1 to the current output device (controlled by CHAN). Each screen listed is followed by a FORM command which is recognized by most printers.

FORM (--) Emit a formfeed character (hex $0C) to the output device. Moves printer to start of next page.

PRINT (-- t;) Any output caused by the command line t will go to the serial port for the printer. EXAMPLE:

PRINT 1 LIST FORM 23 . <ENTER>

This will LIST screen #1 to the serial printer, send a formfeed character, and then print the number 23 .

After all commands up to the <ENTER> key have been executed, the serial port will be turned off and the output res tored to the crt. Before PRINT may be used, the baud rate must be set by one of the BRxxx words.

BR110
BR300
BR600
BR1200
BR2400 (--) Set the serial port baud rate as specified.

PAGE (--) Equivalent to the sequence 1 CLS . It just clears the screen to all green and homes the cursor to the upper left corner.

?CR (--) Tests the system variable OUT for a value greater than 25; if it is, then it does a CR.

9.6 JOYSTICK WORDS:

JSTK (--) Reads all of the joystick values and updates their respective memory locations. Joystick values may be retrieved from their locations by using the words J0, J1, J2, J3.

J0,J1,J2,J3 (-- n) Returns the values of the respective joystick variable. Must first execute JSTK to read them all. E.G.

JSTK J0 J1

Leaves the values of J0 and J1 on the stack. Later, J3 would leave the value of that joystick reading as of the last execution of JSTK.

9.7 TIMER WORDS:

TIME (-- a1) Leaves as a1 the address of a memory location which is decremented by the system clock. The location is decremented each 1/60th of a second, a "tick".

TIMEO (--) Sets the contents of TIME to $FFFF so that a subsequent execution of ?TIME will be able to get the elapsed time in number of "ticks".

?TIME (-- n1) Leaves as n1 the number of "ticks" since the last time TIMEO was executed. It does this by subtracting the current value of TIME from $FFFF. EXAMPLE:

```
: TEST TIMEO 3000 0 DO LOOP ?TIME . ;
```

followed by

```
TEST
```

will print the number of "ticks" of 1/60th of a second to execute the DO LOOP 3000 times.

NOTE: The timer uses the system's 60 hz interrupts, the same ones used by the keyboard input routines. Consequently, do not do a timing operation on any which involves keyboard input.

TICKS (n1 --) System delay routine. n1 specifies a number of 1/60th second "ticks" to wait for. TICKS must be preceded by TIMEO to set the TIMER. The word TICKS will pause until n1 "ticks" have elapsed since the last execution of TIMEO.

EXAMPLES: Suppose that the words T5 and T40 take respectively 5 and 40 "ticks" to execute. Then

```
TIMEO T5 60 TICKS
```

and

```
TIMEO T40 60 TICKS
```

will each take 60 "ticks" to execute. In the example with T5, 60 TICKS will take 55 "ticks"; in the case with T40, 60 TICKS will take only the remaining 20 "ticks".

If you were to type

```
TIMEO T40 T40 60 TICKS
```

then TICKS would return immediately.

TICKS is useful to synchronize parts of programs so that they take equal amounts of time.

9.8 ASSEMBLER WORDS:

CODE (-- t;) Starts a CODE definition. Creates a dictionary header with the name t . It then SMUDGES it, saves the current stack pointer in CSP, and then selects the ASSEMBLER vocabulary.

C; (--) Terminates a CODE definition, verifies that the current stack pointer is equal to the value saved in CSP by CODE, unSMUDGEs the header, and restores the vocabulary to that prior to when the CODE definition was started.

NEXT (--) An ASSEMBLER macro which appends the code for the NEXT operation to the end of a CODE definition. E.g.

```
CODE XXXX x y z . . . NEXT C;
```

will define the word XXXX to perform the assembler code x y z . . . followed by the code for NEXT which returns the execution to the FORTH system, and then C; completes the definition by restoring pointers and checking for stack errors.

9.9 MISCELLANEOUS:

MESS (n1 --) A new definition of the FORTH MESSAGE word. This word prints a brief text explanation instead of just an error number.

SMOVE (a1\a2\n3 --) Does the same function as CMOVE except that it translates the data so that it is displayed properly.

ASCII (-- t; n) Accepts the following string t, takes the first character and leaves its ASCII numerical value on the stack. When compiling, the numerical value is compiled as an inline literal E.g.,

```
ASCII A H.
```

will print 41 , the hex numerical value for A.

9.10 FORTH-79 WORDS:

The following FORTH-79 words have been added to the COLORFORTH system so that it is more compatible with the version of FORTH described by the book STARTING FORTH. Their definitions may be found in that book or in the definition of the FORTH-79 standard which is available from the FORTH INTEREST GROUP. Note: The FORTH-79 STANDARD is the efforts of that group to eliminate the previous differences between various implementations of FORTH. The COLORFORTH system has extensions which make it very close to FORTH-79; however, you should consult the section which discusses the differences if you wish to run FORTH-79 standard programs. For the expert, the FORTH INTEREST GROUP publishes a document describing how to convert fig-FORTH into FORTH-79. The following words in COLORFORTH do most of that conversion already.

?DUP	NEGATE	>IN	EXIT	NOT
CONVERT	DNEGATE	DEPTH	D-	1-
DO=	D<	U<	2@	2!
D=	0>	2CONSTANT	2VARIABLE	DMAX
DMIN	R@	I'	J	

Any words which have different definitions in fig-FORTH than in FORTH-79 have been installed as fig-FORTH. All of the above words are in high level forth, except I' and J.

10.0 FORTH-79 DIFFERENCES

This is a fig-FORTH implementation of the language. However, to ease the transition to FORTH-79, many FORTH-79 words have been added where they do not conflict with fig-FORTH. The differences between COLORFORTH and the STARTING FORTH book will be covered here, on a chapter by chapter basis.

STARTING FORTH	comments
Chapters 1 & 2	No differences noted.

Chapter 3

pg. 60 -- LOAD -- The screens in COLORFORTH are numbered 1 to 8.

pg. 63-88 -- COLORFORTH contains this EDITOR plus many extensions.

pg. 68 -- LINE LENGTH -- The line length in COLORFORTH is 32 characters. So, some of the examples in the book will have lines which are too long. Just break them into two lines. COLORFORTH has 32 lines instead of 16 lines.

pg. 76 -- FLUSH -- The word FLUSH is not needed in this cassette version.

pg. 83 -- HANDY HINTS -- The word DEPTH and .S already exist in COLORFORTH.

Chapter 4

pg. 101 -- ABORT" -- COLORFORTH does not support ABORT" . Just explicitly print out a message with ." and then use ABORT.

Chapter 5 No differences noted.

Chapter 6

pg. 140 -- U.R -- COLORFORTH does not have U.R . You can use .R usually, or if you need an unsigned right justified here is the definition:

```
: U.R   0 SWAP  D.R  ;
```

Chapter 7

pg. 161 -- /LOOP -- Not supported in COLORFORTH.

pg. 162 -- OCTAL -- Not used in COLORFORTH. For most computers, including 6809 systems, HEX is much more natural to use than OCTAL.

pg. 166 -- DOUBLE NUMBER DELIMITERS -- COLORFORTH recognizes only the decimal point as a double number delimiter.

pg. 173 -- DU< -- Not supported in COLORFORTH.

pg. 174 -- M+ and M*/ -- Not supported in COLORFORTH.

Chapter 8

pg. 183 -- VARIABLE -- The definition of VARIABLE used in COLORFORTH is the one from fig-FORTH, which requires an initial value to be on stack before creating the variable. E.g.

```
12 VARIABLE DATE
```

This will create a variable with an initial value of 12. The defintion of 2VARIABLE also requires an initial value on the stack, in this case a double number.

pg. 193 -- 2CONSTANT and 2VARIABLE -- Both are supported in COLORFORTH, but 2VARIABLE requires a double number on the stack for its initial value.

pg. 199 -- ERASE -- This is in COLORFORTH.

pg. 204 -- DUMP -- This is in COLORFORTH.

pg. 207 -- CREATE -- This word functions differently in fig-FORTH than in FORTH-79, and COLORFORTH uses the fig-FORTH definition. Use the following to create the definition of LIMITS as shown in the book:

```
220 VARIABLE LIMITS  340 , 170 , 100 , 190 ,
```

The rule of thumb is to use VARIABLE in place of CREATE for definitions which do NOT have DOES> in them. If the STARTING FORTH book definition is of the form

```
...   CREATE   xxxx  DOES>  xxxxx
```

then use

```
...   <BUILDS  xxxx  DOES>   xxxxx
```

in COLORFORTH. This conforms to the normal fig-FORTH useage.

Chapter 9

pg. 216 -- FIND and EXECUTE -- COLORFORTH uses the fig-FORTH word -FIND in place of FIND. In fig-FORTH the word EXECUTE must receive the code field address instead of the parameter field address. Consequently, on this page of STARTING FORTH, one must change the example to

```
' GREET  CFA  EXECUTE <ENTER>     HELLO I SPEAK FORTH  ok
```

pg. 217 -- VECTORED EXECUTION -- The techniques will work on COLORFORTH with the modification that the addresses obtained with ' are converted to code field addresses by use of CFA. E.g.,

line 6 would read ' HELLO CFA 'ALOHA !

pg. 217 -- SAY -- The definition of SAY in COLORFORTH is

```
: SAY [COMPILE] ' CFA 'ALOHA ! ;
```

There are two changes here. The word ' (tick) is used, and because in fig-FORTH it is immediate, it must be compiled by the [COMPILE] word. The second change is the use of CFA to prepare the address for EXECUTE.

pg. 219 -- NUMBER -- It is not vectored in COLORFORTH, so the example will not work.

pg. 220 -- NAME LENGTHS -- Names in COLORFORTH may be up to 31 characters in length.

pg. 232 -- RELOAD -- No need, all code in rom!

pg. 237 -- H -- In COLORFORTH this word is called DP.

pg. 239 -- OPERATOR -- Not in COLORFORTH.

pg. 240 -- OFFSET -- Not in this cassette system.

pg. 243 -- ASSEMBLER -- COLORFORTH has a minimal ASSEMBLER vocabulary sufficient to hand assemble CODE definitions.

pg. 245 -- LOCATE -- COLORFORTH has the decompiler word SOURCE which may be used to see how a word is defined.

Chapter 10

pg. 255-7 -- UPDATE, FLUSH, SAVE-BUFFERS, EMPTY-BUFFERS, BUFFER These words not in cassette based COLORFORTH.

pg. 259 -- LABEL -- In COLORFORTH must change to

```
: LABEL 8 * ' "LABEL" 3 + + 8 TYPE SPACE ;
```

COLORFORTH does not support ['] and the word ' serves the same function in a definition.

pg. 261 -- >TYPE -- COLORFORTH does not need >TYPE.

pg. 266 -- MOVE and <CMOVE -- Not in COLORFORTH.

pg. 272 -- H -- Use DP .

pg. 281 -- Note: -TEXT in COLORFORTH is different from that defined in STARTING FORTH; see EDITOR definitions.

Chapter 11

pg. 291 -- VARIABLE CREATE -- To create the STARTING FORTH type of definition for VARIABLE, you can do this:

```
: VARIABLE <BUILDS 2 ALLOT DOES> ;
```

To create the COLORFORTH (i.e., fig-FORTH) type of definition for VARIABLE, you can do this:

```
: VARIABLE <BUILDS , DOES> ;
```

The COLORFORTH word CREATE is used only for creating CODE word headers.

pg. 292 -- DEFINING-WORD -- The definition of a DEFINING-WORD in the book must be changed to the following in COLORFORTH:

```
: DEFINING-WORD   <BUILDS    (compile-time action)
                  DOES>      (run-time action)     ;
```

The example for CONSTANT is then

```
: CONSTANT <BUILDS , DOES> @ ;
```

pg. 297 -- ARRAY -- The definition of ARRAY must be changed to

```
: ARRAY <BUILDS OVER , * ALLOT
          DOES> DUP @ ROT * + + 2+ ;
```

pg. 313 -- DOES> -- For most purposes COLORFORTH is the same as FORTH-79, but for the advanced programmer, see the document FORTH-79 STANDARD CONVERSION from the FORTH INTEREST GROUP.

pg. 332 -- JOB 1FIELD 2FIELD -- Again, these definitions must be changed to account for the different way CREATE works. Use

```
20 VARIABLE JOB    24 ,
00 VARIABLE 1FIELD 30 ,
30 VARIABLE 2FIELD 12 ,
```

pg. 339 -- SIMPLE FILES -- In screen 240 change the definitions of SURNAME, GIVEN, JOB, PHONE to the following:

```
00 VARIABLE SURNAME  16 ,
16 VARIABLE GIVEN    12 ,
28 VARIABLE JOB      24 ,
52 VARIABLE PHONE    12 ,
```

pg. 339 -- FREE -- Change the definition of FREE to the following:

```
: FREE  1 MAXRECS  0
        DO  I OECORD   !   RECORD C@  33 <
                IF NOT LEAVE THEN
        LOOP
        IF  ." FILE FULL "  ABORT  THEN ;
```

pg. 339 -- ' (tick) -- Prefix all occurrences of ' with the word [COMPILE] , e.g.,

```
: CHANGE  [COMPILE] ' PUT ;
```

pg. 347 -- screens 246 and 248 -- The definitions of DENSITY, THETA, and STRING will need to be prefixed with a ZERO (0) .

11.0 ERRORS, CRASHES, AND OTHER SUCH PROBLEMS

11.1 CRASHES

In the process of writing programs, we all make mistakes now and then. When this happens, there is a good chance that we will "crash" the program. If this happens, do not dispair! Since COLORFORTH is in rom, it does not get wiped out; it is simple to recover.

DO NOT TURN OFF THE POWER, as this will cause your source code in the BLOCK buffers to be lost. Just press the RESET button on the right rear. FORTH will reinitialize with the normal sign on message. FORTH does NOT clear out any buffer space on initialization, although it does reset BMAX to 8 (1 on a 4k system). So, if your program crash did not write over the buffer area, your program is preserved. You can go back to it to figure out the problem, reload, and retry it. First list out the source screens and look for bad code. If you find mistakes, fix them by editing. You may find that some "garbage" is in the screens, presumably because when your program went west, it wrote into the buffers. In this case, you can edit the errors, and go on.

11.2 ERROR MESSAGES

When COLORFORTH detects an error, it will stop compiling and an appropriate error message is issued. The following is a list of the errors and their interpretation.

ERROR MESSAGE	MEANING
WHAT	FORTH could not f ind the word in the dictionary
STACK EMPTY	Some word tried to take something from the stack after it was empty.
MEM FULL	You have used up all available memory.
REDEF:	You have just redefined a word with a name which already exists in the dictionary. This is not a fatal error, just an informational message. Sometimes one deliberately redefines a name. Sometimes this reveals a mistake.
BLK RANGE	Attempted to access a BLOCK number outside the allowed range (normally 1 - 8).
?COMPILE	Tried to execute a word that may be used only within a definition.
?EXECUTE	Tried to compile a word into a definition which was meant for execution only.

?PAIRS	A conditional structure such as IF ... ENDIF or DO ... LOOP was used without the correct matching terminating word.
NOT DONE	Attempted to terminate a definition with a ; before you finished it.
SAVED VOC	You have attempted to FORGET a word in the permanent rom area or below FENCE.
?LOADING	Have attempted to execute a word that applies applies to loading only.
OFFSCREEN	During an edit session the cursor position was found to point outside of the current screen.
SET VOCAB	You tried to FORGET a word from a vocabulary other than the one you are currently in. You need to force the vocabulary pointers to the correct one by stating correct-vocabulary-name DEFINITIONS
BAD OVERLAY	During use of an overlay word, ?OV looked at the program in memory and found it to not be an overlay.

12.0 SOME EXAMPLE CODE AND HANDY UTILITIES

12.1 A JOYSTICK EDITOR

This addition to the EDITOR is given in source code form, in the listings on screens 23 and 24. You can type them in (they may go into any screen). This editor allows you to use the joystick to move the cursor around in the screen and then you can manipulate the text more easily. A very handy addition is the word GET which picks up the word under the cursor so that you can INSERT it later somewhere else.

The best procedure is to use the regular editor to place this into some screens, and then save it on cassette tape for later use. Then you can start it up by typing

```
n1  EDIT   J  <ENTER>
```

where the number n1 is the screen number you wish to edit. You can now move the JOYSTICK around to move the cursor. CAUTION!! Many of the keys on the keyboard have an immediate function, i.e., you do not have to press <ENTER> after each key. Each of the old commands of the regular editor (e.g., P, X, etc.) has been assigned to a key -- see the source screen for J for the definitions). Pressing that key alone (no <ENTER>) will cause that word to be executed. For example, to insert some text at the position pointed to by the cursor, press I . The black cursor stops flashing and you are prompted with:

```
>
```

You should enter the string of text you wish to insert, followed by <ENTER>. When you press <ENTER> the text appears at the position of the cursor and control returns to the JOYSTICK.

Try it! E.g., insert this somewhere:

```
THIS IS A TEST
```

Now, suppose that you wish to move the second word of the above sentence to the next line. Position the cursor over the I or S in the word IS and then press the W button. The word disappears and the text on the line is compressed. The word IS is now in PADI, so we can insert it elsewhere. Move the cursor down one line, place it in the space just preceding the T in THIS on the previous line. Now press the I key, followed by <ENTER> (since the text to insert is already at PADI). The text will be placed in the new line!

This is only a small sample of what you can do. Try doing a sequence of things such as H followed by I . The N and B keys allow you to move to the next higher or lower screen for editing. You can do anything you want! You have the source!

12.2 PRINTING

A couple of useful utilities are given in screen 22 of the source listing. The words DLIST and LTHRU allow you to print pairs of screens on a single page as was done in the documentation. DLIST was used as follows:

```
PRINT 1 DLIST <ENTER>
```

This will list screens 1 and 2 to the serial port to a printer, side by side. The word LTHRU is similar to THRU, except it is used for printing as

```
PRINT 1 6 LTHRU <ENTER>
```

Screens 1 through 6 will be printed in the DLIST format. Be sure to set the BAUD rate before using PRINT.

12.3 WORD LISTS

The fig-FORTH word VLIST gives a listing of the words in the system. It types them in the order in which they are found, the inverse order in which they were entered.

The following word may be typed in and it will give you a listing sorted according to the ASCII order of the first character in the name:

```
HEX
  : ALIST 80 20 DO
                    CONTEXT @ @
                    BEGIN   DUP 1+ C@    7F AND   I =
                            IF CR DUP    PFA H. SPACE    DUP ID.
                            ENDIF        PFA LFA @       DUP 0=
                    UNTIL   DROP ?TERMINAL IF LEAVE ENDIF
            LOOP   ;
```

12.4 ALTERNATE COLORS

The COLOR COMPUTER in one of the high resolution graphics modes can display one of two sets of four colors. To switch to the other set, one must change the fourth bit in the location hex $FF22. A word to do this is

```
HEX  : ALT  FF22 8 TOGGLE ;
```

13.0 COMMENTS ON THE SOURCE LISTING

This entire FORTH system was generated on a RADIO SHACK COLOR COMPUTER with 32k bytes of memory. While 32k may sound like a lot of memory, when you are developing a program which is 10k in length without benefit of disk, 32k is minimal. You will notice that the source is broken into two seperate types of listings: the assembly listing and the high level FORTH source code. This is because it was possible to hold in the computer enough source for only 5k of machine code. The next step was to type in a minimal, very small editor from the keyboard. Once that was in, it was used to enter a full editor into screens and they were saved on cassette for later use.

Now with a useful editor, it was easier to work! The rest of the COLORFORTH system was gradually developed until a 10k system was complete. The final step was to patch the bootup dictionary links at C00E and C010 so that the full COLORFORTH would come up upon cold start. That was burned into proms.

We would like to apologize for the lack of comments in the listing. It was caused by a lack of space. There was just barely room to get the 5k of assembly code in, much less comments.

Some interesting notes about the high level source: There are several words in the listing which do not appear in a VLIST. This is because their names in the dictionary were overwritten with a space. The first of these is on screen 1, line 18. The word 'B' was created only to be used to create the words for doing Baud rate setting. After it was used to create BR110 -- BR2400, its function was no longer needed, so it was deleted in line 24. This technique was used several time throughout the source. We were not trying to hide anything from you (you have the source), but rather we were trying to keep the dictionary from getting cluttered with single character, meaningless names. We could have given them longer names, but that would just use up valuable space for things you would never use.

In screen 5, lines 19 to 22, the words 'J' and 'JJ' appear. We wanted to put in down arrows, but control characters are not always visible. So we compiled the words with letter names and then in lines 21 and 23 they were written over with down arrows (Just try that with a FORTRAN compiler! or a BASIC interpreter!).

It is interesting to note that 19 pages of FORTH creates about 5k of compiled code, while it took over 30 dense pages of assembly to create 5k of code. FORTH is much more efficient and much more easily readable.

We hope that you find this an enjoyable product. We enjoyed doing it, and it is our hope that you will find that FORTH is a useful powerful language for getting more out of your COLOR COMPUTER. If you write programs which run with COLORFORTH, consider marketing them. FORTH is an excellent language for writting games and doing graphics. The benchmarks we have done show that COLORFORTH is much faster than COLOR BASIC, so that you can do games and such in real time with high level code.

Future products being planned are DISK versions of COLORFORTH and a TINY PASCAL compiler running on COLORFORTH. What are you going to write?

The present source code could be adapted for use on other 6809 computers. However, much of it is specific to the COLOR COMPUTER, and other parts are done in a space-conserving fashion so that we could get a maximum of utility into a given space of ROMs. If you have other 6809 systems, contact TALBOT MICROSYSTEMS for information on its line of tFORTH products for SS-50 bus and EXORCISER(tm Motorola) bus systems.

```
001 0600                        NAM COLORF
                              *COPYRIGHT 1981
                              *TJ ZIMMER & RJ TALBOT
002 0600                      PRGBGN EQU $C000
003 0600                      BRAM   EQU $0620
004 0600                      NBLK   EQU 8
005 0600                      USRBGN EQU BRAM+$80    CAUTION!!
006 0600                      USREND EQU USRBGN+$300
007 0600                      VIRBGN EQU USREND
008 0600                      VIREND EQU VIRBGN+1028*NBLK
009 0600                      N      EQU USRBGN
010 0600                      UP     EQU USRBGN+$0A
011 0600                      UORIG  EQU USRBGN+$0C
012 0600                             ORG PRGBGN
013 C000 160127               KRNL   LBRA CENT
014 C003 160173                      LBRA WENT
015 C006 06AC                 UPINIT FDB UORIG
016 C008 067B                 FENCIN FDB ERAM-RAM+BRAM
017 C00A 29C0                 DPINIT FDB VIREND
018 C00C 064E                 VOCINT FDB FORTH+8-RAM+BRAM
019 C00E 0666                 TOPDEF FDB EDITOR-9-RAM+BRAM
020 C010 064A                 TOPEDT FDB FORTH+4-RAM+BRAM
021 C012 0008                 MAXBLK FDB NBLK
022 C014 0000                        FDB $00
023 C016 09A0                 XVIRBG FDB VIRBGN
024 C018 29C0                 XVIRED FDB VIREND
025 C01A 08A0                 SINIT  FDB USREND-$100
026 C01C 08A0                 TIBINT FDB USREND-$100
027 C01E 09A0                 RINIT  FDB USREND
028 C020 00000000                    FDB 0,0
029 C024 001F                 WIDINT FDB 31
030 C026 0000                        FDB 0
031 C028 0000                 WRNINT FDB 0
032 C02A 3706                 PULLDX PULU D
033 C02C ED84                 STOREX STD ,X
034 C02E 2022                        BRA NEXT
035 C030 EC84                 GETX   LDD ,X
036 C032 3606                 PUSHD  PSHU D
037 C034 201C                        BRA NEXT
038 C036 C1BA                        FCB $C1,':+$80
039 C038 0000                        FDB 0
040 C03A C04ECA68CA           COLON  FDB DOCOL,QEXEC,SCSP
041 C040 C897C6C0C8                  FDB CURENT,AT,CONTXT
042 C046 C6D8CF3BCA                  FDB STORE,CREATE,RBRAK
043 C04C CB40                        FDB PSCODE
044 C04E 3420                 DOCOL  PSHS Y
045 C050 3102                        LEAY 2,X
046 C052 AEA1                 NEXT   LDX ,Y++
047 C054 6E94                 NEXT3  JMP [,X]
048 C056 823BD3                      FCB $82,';,'S+$80
049 C059 C036                        FDB COLON-4
050 C05B C05D                 SEMIS  FDB *+2
```

```
051 C05D 10AEE1       PSEMIS LDY ,S++
052 C060 20F0                BRA NEXT
053 C062 87                  FCB $87
054 C063 4558454355          FCC /EXECUT/
055 C069 C5                  FCB 'E+$80
056 C06A C056                FDB SEMIS-5
057 C06C C06E         EXEC   FDB *+2
058 C06E 3710                PULU X
059 C070 20E2                BRA NEXT3
060 C072 852D544558          FCB $85,'-,'T,'E,'X
061 C077 D4                  FCB 'T+$80
062 C078 C062                FDB EXEC-10
063 C07A C07C         MTEXT  FDB *+2
064 C07C 3430                PSHS X,Y
065 C07E 3710                PULU X
066 C080 3726                PULU D,Y
067 C082 A680         MTEXT2 LDA  ,X+
068 C084 A1A0                CMPA ,Y+
069 C086 2607                BNE NMTCH
070 C088 5A                  DECB
071 C089 26F7                BNE MTEXT2
072 C08B C601                LDB #$01
073 C08D 2001                BRA MEXIT
074 C08F 5F           NMTCH  CLRB
075 C090 4F           MEXIT  CLRA
076 C091 3530                PULS X,Y
077 C093 16FF9C              LBRA PUSHD
078 C096 84454D49D4          FCB $84,'E,'M,'I,'T+$80
079 C09B C072                FDB MTEXT-8
080 C09D C04EC6A3C1   EMIT   FDB DOCOL,DUP,LIT,7,EQUAL
081 C0A7 C1E6                FDB ZBRAN
082 C0A9 000AC1CA00          FDB EMIT2-*,LIT,238
083 C0AF C78CD47E            FDB ONE,TONE
084 C0B3 C0B7C05B     EMIT2  FDB CEMIT,SEMIS
085 C0B7 C0B9         CEMIT  FDB *+2
086 C0B9 3706                PULU D
087 C0BB 3434                PSHS B,X,Y
088 C0BD 1F98                TFR B,A
089 C0BF AD9FA002            JSR [$A002]
090 C0C3 3534                PULS B,X,Y
091 C0C5 BE06D6              LDX UORIG+$2A
092 C0C8 3001                LEAX 1,X
093 C0CA BF06D6              STX UORIG+$2A
094 C0CD 16FF82              LBRA NEXT
095 C0D0 834B45D9            FCB $83,'K,'E,'Y+$80
096 C0D4 C096                FDB EMIT-7
097 C0D6 C04EC0E2C7   KEY    FDB DOCOL,CKEY,ZERO
098 C0DC C784D47EC0          FDB ZERO,TONE,SEMIS
099 C0E2 C0E4         CKEY   FDB *+2
100 C0E4 3430                PSHS X,Y
101 C0E6 BDA1B1              JSR  $A1B1
102 C0E9 1F89                TFR A,B
```

```
103 C0EB 3530                PULS X,Y
104 C0ED 4F                  CLRA
105 C0EE 16FF41              LBRA PUSHD
106 C0F1 89                  FCB $89
107 C0F2 3F5445524D          FCC /?TERMINA/
108 C0FA CC                  FCB 'L+$80
109 C0FB C0D0                FDB KEY-6
110 C0FD C0FF        0TERM   FDB *+2
111 C0FF 3430                PSHS X,Y
112 C101 AD9FA000            JSR  [$A000]
113 C105 3530                PULS X,Y
114 C107 1F89                TFR A,B
115 C109 4F                  CLRA
116 C10A 16FF25              LBRA PUSHD
117 C10D 8243D2              FCB $82,'C,'R+$80
118 C110 C0F1                FDB 0TERM-$0C
119 C112 C04ECBE4    CR      FDB DOCOL,PDOTQ
120 C116 020D0A              FCB 2,13,10
121 C119 C784C864            FDB ZERO,OUT
122 C11D C6D8C05B            FDB STORE,SEMIS
123 C121 84434F4CC4          FCB $84,'C,'O,'L,'D+$80
124 C126 C10D                FDB CR-5
125 C128 C12A        COLD    FDB *+2
126 C12A 4F          CENT    CLRA
127 C12B 1F8B                TFR A,DP
128 C12D 1CEF                ANDCC #$EF
129 C12F CE067B              LDU #ERAM-RAM+BRAM
130 C132 8ED5CB              LDX #ERAM
131 C135 A682        COLD2   LDA ,-X
132 C137 A7C2                STA ,-U
133 C139 8CD570              CPX #RAM
134 C13C 26F7                BNE COLD2
135 C13E BEC00E              LDX TOPDEF
136 C141 BF064C              STX FORTH+6-RAM+BRAM
137 C144 BEC010              LDX TOPEDT
138 C147 BF0675              STX EDITOR+6-RAM+BRAM
139 C14A 10FEC01E            LDS RINIT
140 C14E CE06C2              LDU #UORIG+$16
141 C151 8EC016              LDX #XVIRBG
142 C154 A682        COLDZ   LDA ,-X
143 C156 A7C2                STA ,-U
144 C158 8CC008              CPX #FENCIN
145 C15B 26F7                BNE COLDZ
146 C15D 9674                LDA $74
147 C15F 8130                CMPA #$30
148 C161 2216                BHI WENT
149 C163 8601                LDA #$01
150 C165 B706BF              STA UORIG+$13
151 C168 CC0DA4              LDD #$0DA4
152 C16B FD06B6              STD UORIG+$0A
153 C16E 2009                BRA WENT
154 C170 84574152CD          FCB $84,'W,'A,'R,'M+$80
```

changed to ØØ

```
0155 C175 C121                FDB COLD-7
0156 C177 C179          WARM  FDB *+2
0157 C179 CE06D2        WENT  LDU #UORIG+$26
0158 C17C 8EC02A              LDX #WRNINT+2
0159 C17F A682          WARM2 LDA ,-X
0160 C181 A7C2                STA ,-U
0161 C183 8CC01A              CPX #SINIT
0162 C186 26F7                BNE WARM2
0163 C188 FEC01A              LDU SINIT
0164 C18B BEC006              LDX UPINIT
0165 C18E BF06AA              STX UP
0166 C191 108ED0DE            LDY #ABORT+2
0167 C195 160025              LBRA RPSTOR+2
0168 C198 835350C0            FCB $83,'S,'P,'@+$80
0169 C19C C170                FDB WARM-7
0170 C19E C1A0          SPAT  FDB *+2
0171 C1A0 30C4                LEAX ,U
0172 C1A2 3610                PSHU X
0173 C1A4 16FEAB              LBRA NEXT
0174 C1A7 835350A1            FCB $83,'S,'P,'!+$80
0175 C1AB C198                FDB SPAT-6
0176 C1AD C1AF          SPSTOR FDB *+2
0177 C1AF FEC01A              LDU SINIT
0178 C1B2 16FE9D              LBRA NEXT
0179 C1B5 835250A1            FCB $83,'R,'P,'!+$80
0180 C1B9 C1A7                FDB SPSTOR-6
0181 C1BB C1BD          RPSTOR FDB *+2
0182 C1BD 10FEC01E            LDS RINIT
0183 C1C1 16FE8E              LBRA NEXT
0184 C1C4 834C49D4            FCB $83,'L,'I,'T+$80
0185 C1C8 C1B5                FDB RPSTOR-6
0186 C1CA C1CC          LIT   FDB *+2
0187 C1CC ECA1                LDD ,Y++
0188 C1CE 16FE61              LBRA PUSHD
0189 C1D1 864252414E          FCB $86,'B,'R,'A,'N,'C
0190 C1D7 C8                  FCB 'H+$80
0191 C1D8 C1C4                FDB LIT-6
0192 C1DA C1EC          BRAN  FDB ZBYES
0193 C1DC 8730425241          FCB $87,'0,'B,'R,'A,'N,'C
0194 C1E3 C8                  FCB 'H+$80
0195 C1E4 C1D1                FDB BRAN-$09
0196 C1E6 C1E8          ZBRAN FDB *+2
0197 C1E8 ECC1                LDD ,U++
0198 C1EA 2609                BNE ZBNO
0199 C1EC 1F20          ZBYES TFR Y,D
0200 C1EE E3A4                ADDD ,Y
0201 C1F0 1F02                TFR D,Y
0202 C1F2 16FE5D              LBRA NEXT
0203 C1F5 3122          ZBNO  LEAY 2,Y
0204 C1F7 16FE58              LBRA NEXT
0205 C1FA 86284C4F4F          FCB $86,'(,'L,'O,'O,'P
0206 C200 A9                  FCB ')+$80
```

```
207 C201 C1DC                FDB ZBRAN-$0A
208 C203 C205       XLOOP    FDB *+2
209 C205 CC0001              LDD #$01
210 C208 200E                BRA XPLOP2
211 C20A 87282B4C4F          FCB $87,'(,'+,'L,'O,'O,'P
212 C211 A9                  FCB ')+$80
213 C212 C1FA                FDB XLOOP-9
214 C214 C216       XPLOOP   FDB *+2
215 C216 3706                PULU D
216 C218 4D         XPLOP2   TSTA
217 C219 2A0E                BPL XPLOF       FOR
218 C21B E3E4                ADDD ,S
219 C21D EDE4                STD ,S
220 C21F 1C01                ANDCC #$01
221 C221 E263                SBCB 3,S
222 C223 A262                SBCA 2,S
223 C225 2AC5                BPL ZBYES
224 C227 2008                BRA XPLONO
225 C229 E3E4       XPLOF    ADDD ,S         BACK
226 C22B EDE4                STD ,S
227 C22D A362                SUBD 2,S
228 C22F 2BBB                BMI ZBYES
229 C231 3264       XPLONO   LEAS 4,S
230 C233 20C0                BRA ZBNO
231 C235 8428444FA9          FCB $84,'(,'D,'O,')+$80
232 C23A C20A                FDB XPLOOP-$0A
233 C23C C23E       XDO      FDB *+2
234 C23E 3706                PULU D
235 C240 3710                PULU X
236 C242 3416                PSHS X,D
237 C244 16FE0B              LBRA NEXT
238 C247 81C9                FCB $81,'I+$80
239 C249 C235                FDB XDO-7
240 C24B C24D       I        FDB *+2
241 C24D ECE4                LDD ,S
242 C24F 16FDE0              LBRA PUSHD
243 C252 8544494749          FCB $85,'D,'I,'G,'I
244 C257 D4                  FCB 'T+$80
245 C258 C247                FDB I-4
246 C25A C25C       DIGIT    FDB *+2
247 C25C A643                LDA 3,U
248 C25E 8030                SUBA #$30
249 C260 2B1B                BMI DIGIT2      BAD<0
250 C262 810A                CMPA #$0A
251 C264 2B0A                BMI DIGIT0      OK<9
252 C266 8111                CMPA #$11
253 C268 2B13                BMI DIGIT2      BAD<A
254 C26A 812B                CMPA #$2B
255 C26C 2A0F                BPL DIGIT2      BAD>Z
256 C26E 8007                SUBA #$07
257 C270 A141       DIGIT0   CMPA 1,U
258 C272 2A09                BPL DIGIT2      BAD>BASE
```

```
259 C274 C601                LDB #$01
260 C276 A743                STA 3,U
261 C278 E741          DIGIT1 STAB 1,U
262 C27A 16FDD5              LBRA NEXT
263 C27D 5F            DIGIT2 CLRB
264 C27E 3342                LEAU 2,U
265 C280 E7C4                STAB 0,U
266 C282 20F4                BRA DIGIT1
267 C284               PD    EQU N
268 C284               PA0   EQU N+2
269 C284               PA    EQU N+4
270 C284               PCHR  EQU N+6
271 C284 862846494E          FCB $86,'(,'F,'I,'N,'D
272 C28A A9                  FCB ')+$80
273 C28B C252                FDB DIGIT-8
274 C28D C28F          PFIND FDB *+2
275 C28F 3420                PSHS Y
276 C291 3730          PFIND0 PULU X,Y
277 C293 10BF06A2            STY PA0
278 C297 E680          PFIND1 LDB ,X+
279 C299 F706A6              STAB PCHR
280 C29C C43F                ANDB #$3F
281 C29E 10BE06A2            LDY PA0
282 C2A2 E1A0                CMPB 0,Y+
283 C2A4 2618                BNE PFIND4
284 C2A6 A6A0          PFIND2 LDA ,Y+
285 C2A8 6D84                TST ,X
286 C2AA 2A0E                BPL PFIND8
287 C2AC 8A80                ORA #$80
288 C2AE A180                CMPA ,X+
289 C2B0 2712                BEQ FOUND
290 C2B2 AE84          PFIND3 LDX 0,X
291 C2B4 26E1                BNE PFIND1
292 C2B6 1F10                TFR X,D
293 C2B8 2014                BRA PFINDE
294 C2BA A180          PFIND8 CMPA ,X+
295 C2BC 27E8                BEQ PFIND2
296 C2BE E680          PFIND4 LDB ,X+
297 C2C0 2AFC                BPL PFIND4
298 C2C2 20EE                BRA PFIND3
299 C2C4 3004          FOUND  LEAX 4,X
300 C2C6 F606A6              LDB PCHR
301 C2C9 4F                  CLRA
302 C2CA 3616                PSHU X,D
303 C2CC C601                LDB #$01
304 C2CE 3520          PFINDE PULS Y
305 C2D0 16FD5F              LBRA PUSHD
306 C2D3 87454E434C          FCB $87,'E,'N,'C,'L,'O,'S
307 C2DA C5                  FCB 'E+$80
308 C2DB C284                FDB PFIND-9
309 C2DD C2DF          ENCLOS FDB *+2
310 C2DF 3706                PULU D
```

```
311 C2E1 3420              PSHS Y
312 C2E3 8EFFFF            LDX #$FFFF
313 C2E6 10AEC4            LDY ,U
314 C2E9 3001       ENCL01 LEAX 1,X
315 C2EB E1A0              CMPB ,Y+
316 C2ED 27FA              BEQ ENCL01
317 C2EF 3610              PSHU X
318 C2F1 6D3F              TST -1,Y
319 C2F3 2604              BNE ENCL02
320 C2F5 3001              LEAX 1,X
321 C2F7 200A              BRA ENCL11
322 C2F9 3001       ENCL02 LEAX 1,X
323 C2FB E1A4              CMPB ,Y
324 C2FD 2708              BEQ ENCL2
325 C2FF 6DA0              TST ,Y+
326 C301 26F6              BNE ENCL02
327 C303 3610       ENCL11 PSHU X
328 C305 2004              BRA ENCL4
329 C307 3610       ENCL2  PSHU X
330 C309 3001              LEAX 1,X
331 C30B 3610       ENCL4  PSHU X
332 C30D 3520              PULS Y
333 C30F 16FD40            LBRA NEXT
334 C312 85434D4F56        FCB $85,'C,'M,'O,'V
335 C317 C5                FCB 'E+$80
336 C318 C2D3              FDB ENCLOS-$0A
337 C31A C31C       CMOVE  FDB *+2
338 C31C 3430              PSHS X,Y
339 C31E 3736              PULU D,X,Y
340 C320 3440              PSHS U
341 C322 1F23              TFR Y,U
342 C324 1F02              TFR D,Y
343 C326 3121              LEAY 1,Y
344 C328 313F       CMOV2  LEAY -1,Y
345 C32A 2706              BEQ CMOV3
346 C32C A6C0              LDA ,U+
347 C32E A780              STA ,X+
348 C330 20F6              BRA CMOV2
349 C332 3540       CMOV3  PULS U
350 C334 3530              PULS X,Y
351 C336 16FD19            LBRA NEXT
352 C339 85534D4F56        FCB $85,'S,'M,'O,'V
353 C33E C5                FCB 'E+$80
354 C33F C312              FDB CMOVE-8
355 C341 C343       SMOVE  FDB *+2
356 C343 3430              PSHS X,Y
357 C345 3736              PULU D,X,Y
358 C347 3440              PSHS U
359 C349 1F23              TFR Y,U
360 C34B 1F02              TFR D,Y
361 C34D 3121              LEAY 1,Y
362 C34F 313F       SMOV2  LEAY -1,Y
```

```
0363 C351 2710                BEQ SMOV3
0364 C353 A6C0                LDA ,U+
0365 C355 815F                CMPA #$5F
0366 C357 2206                BHI LCASE
0367 C359 8A40                ORA #$40
0368 C35B A780          SMOV4 STA ,X+
0369 C35D 20F0                BRA SMOV2
0370 C35F 841F          LCASE ANDA #$1F
0371 C361 20F8                BRA SMOV4
0372 C363 3540          SMOV3 PULS U
0373 C365 3530                PULS X,Y
0374 C367 16FCE8              LBRA NEXT
0375 C36A 84574156            FCB $84,'W,'A,'V
0376 C36E C5                  FCB 'E+$80
0377 C36F C339                FDB SMOVE-8
0378 C371 C373          WAVE  FDB *+2
0379 C373 3430                PSHS X,Y
0380 C375 3736                PULU D,X,Y
0381 C377 3440                PSHS U
0382 C379 1F23                TFR Y,U
0383 C37B 1F02                TFR D,Y
0384 C37D 9F02                STX $02
0385 C37F 863C                LDA #$3C
0386 C381 B7FF23              STA $FF23
0387 C384 3121                LEAY 1,Y
0388 C386 313F          WAVE2 LEAY -1,Y
0389 C388 270D                BEQ WAVE3
0390 C38A A6C0                LDA ,U+
0391 C38C B7FF20              STA $FF20
0392 C38F 9E02                LDX $02
0393 C391 301F          WAVE4 LEAX -1,X
0394 C393 26FC                BNE WAVE4
0395 C395 20EF                BRA WAVE2
0396 C397 3540          WAVE3 PULS U
0397 C399 3530                PULS X,Y
0398 C39B 16FCB4              LBRA NEXT
0399 C39E 8255AA              FCB $82,'U,'*+$80
0400 C3A1 C36A                FDB WAVE-7
0401 C3A3 C3A5          USTAR FDB *+2
0402 C3A5 3716                PULU D,X
0403 C3A7 3416                PSHS D,X
0404 C3A9 A661                LDA 1,S
0405 C3AB E663                LDB 3,S
0406 C3AD 3D                  MUL
0407 C3AE 3606                PSHU D
0408 C3B0 4F                  CLRA
0409 C3B1 5F                  CLRB
0410 C3B2 3606                PSHU D
0411 C3B4 A6E4                LDA 0,S
0412 C3B6 E663                LDB 3,S
0413 C3B8 3D                  MUL
0414 C3B9 E341                ADDD 1,U
```

```
0415 C3BB ED41                STD 1,U
0416 C3BD A661                LDA 1,S
0417 C3BF E662                LDB 2,S
0418 C3C1 3D                  MUL
0419 C3C2 E341                ADDD 1,U
0420 C3C4 ED41                STD 1,U
0421 C3C6 A6E4                LDA 0,S
0422 C3C8 E662                LDB 2,S
0423 C3CA 3D                  MUL
0424 C3CB E3C4                ADDD 0,U
0425 C3CD EDC4                STD 0,U
0426 C3CF 3264                LEAS 4,S
0427 C3D1 16FC7E              LBRA NEXT
0428 C3D4 8255AF              FCB $82,'U,'/+$80
0429 C3D7 C39E                FDB USTAR-5
0430 C3D9 C3DB        USLASH  FDB *+2
0431 C3DB EC42                LDD 2,U
0432 C3DD AE44                LDX 4,U
0433 C3DF AF42                STX 2,U
0434 C3E1 ED44                STD 4,U
0435 C3E3 6843                ASL 3,U
0436 C3E5 6942                ROL 2,U
0437 C3E7 8E0010              LDX #$10
0438 C3EA 6945        USLL1   ROL 5,U
0439 C3EC 6944                ROL 4,U
0440 C3EE EC44                LDD 4,U
0441 C3F0 A3C4                SUBD ,U
0442 C3F2 1CFE                ANDCC #$FE      CLC
0443 C3F4 2B04                BMI USLL2
0444 C3F6 ED44                STD 4,U
0445 C3F8 1A01                ORCC #1         SEC
0446 C3FA 6943        USLL2   ROL 3,U
0447 C3FC 6942                ROL 2,U
0448 C3FE 301F                LEAX -$1,X
0449 C400 26E8                BNE USLL1
0450 C402 3342                LEAU 2,U
0451 C404 16FC4B              LBRA NEXT
0452 C407 8232AA              FCB $82,'2,'*+$80
0453 C40A C3D4                FDB USLASH-5
0454 C40C C40E        TSTR    FDB *+2
0455 C40E 6841                LSL 1,U
0456 C410 69C4                ROL 0,U
0457 C412 16FC3D              LBRA NEXT
0458 C415 8232AF              FCB $82,'2,'/+$80
0459 C418 C407                FDB TSTR-5
0460 C41A C41C        TSLS    FDB *+2
0461 C41C 67C4                ASR 0,U
0462 C41E 6641                ROR 1,U
0463 C420 16FC2F              LBRA NEXT
0464 C423 83414EC4            FCB $83,'A,'N,'D+$80
0465 C427 C415                FDB TSLS-5
0466 C429 C42B        AND     FDB *+2
```

```
0467 C42B 3706                  PULU D
0468 C42D E441                  ANDB 1,U
0469 C42F A4C4                  ANDA 0,U
0470 C431 EDC4          PUTD    STD ,U
0471 C433 16FC1C                LBRA NEXT
0472 C436 824FD2                FCB $82,'O,'R+$80
0473 C439 C423                  FDB AND-6
0474 C43B C43D          OR      FDB *+2
0475 C43D 3706                  PULU D
0476 C43F EA41                  ORB 1,U
0477 C441 AAC4                  ORA 0,U
0478 C443 20EC                  BRA PUTD
0479 C445 83584FD2              FCB $83,'X,'O,'R+$80
0480 C449 C436                  FDB OR-5
0481 C44B C44D          XOR     FDB *+2
0482 C44D 3706                  PULU D
0483 C44F E841                  EORB 1,U
0484 C451 A8C4                  EORA 0,U
0485 C453 20DC                  BRA PUTD
0486 C455 81AB                  FCB $81,'++$80
0487 C457 C445                  FDB XOR-6
0488 C459 C45B          PLUS    FDB *+2
0489 C45B 3706                  PULU D
0490 C45D E3C4                  ADDD ,U
0491 C45F 16FFCF                LBRA PUTD
0492 C462 8244AB                FCB $82,'D,'++$80
0493 C465 C455                  FDB PLUS-4
0494 C467 C469          DPLUS   FDB *+2
0495 C469 EC42                  LDD 2,U
0496 C46B E346                  ADDD 6,U
0497 C46D ED46                  STD 6,U
0498 C46F ECC4                  LDD ,U
0499 C471 E945                  ADCB 5,U
0500 C473 A944                  ADCA 4,U
0501 C475 3344                  LEAU 4,U
0502 C477 EDC4                  STD ,U
0503 C479 16FBD6                LBRA NEXT
0504 C47C 854D494E55            FCB $85,'M,'I,'N,'U
0505 C481 D3                    FCB 'S+$80
0506 C482 C462                  FDB DPLUS-5
0507 C484 C486          MINUS   FDB *+2
0508 C486 6041                  NEG 1,U
0509 C488 2505                  BCS MINUS2
0510 C48A 60C4                  NEG ,U
0511 C48C 16FBC3                LBRA NEXT
0512 C48F 63C4          MINUS2  COM ,U
0513 C491 16FBBE                LBRA NEXT
0514 C494 86444D494E            FCB $86,'D,'M,'I,'N,'U
0515 C49A D3                    FCB 'S+$80
0516 C49B C47C                  FDB MINUS-8
0517 C49D C49F          DMINUS  FDB *+2
0518 C49F 63C4                  COM 0,U
```

```
0519 C4A1 6341                COM 1,U
0520 C4A3 6342                COM 2,U
0521 C4A5 6043                NEG 3,U
0522 C4A7 260A                BNE DMINX
0523 C4A9 6C42                INC 2,U
0524 C4AB 2606                BNE DMINX
0525 C4AD 6C41                INC 1,U
0526 C4AF 2602                BNE DMINX
0527 C4B1 6CC4                INC ,U
0528 C4B3 16FB9C      DMINX   LBRA NEXT
0529 C4B6 8231AB              FCB $82,'1,'++$80
0530 C4B9 C494                FDB DMINUS-9
0531 C4BB C4BD        ONEP    FDB *+2
0532 C4BD ECC4                LDD ,U
0533 C4BF C30001              ADDD #1
0534 C4C2 16FF6C              LBRA PUTD
0535 C4C5 8232AB              FCB $82,'2,'++$80
0536 C4C8 C4B6                FDB ONEP-5
0537 C4CA C4CC        TWOP    FDB *+2
0538 C4CC CC0002              LDD #2
0539 C4CF E3C4                ADDD ,U
0540 C4D1 16FF5D              LBRA PUTD
0541 C4D4 824DAA              FCB $82,'M,'*+$80
0542 C4D7 C4C5                FDB TWOP-5
0543 C4D9 C04EC676C6  MSTAR   FDB DOCOL,OVER,OVER,XOR
0544 C4E1 C652C59AC6          FDB TOR,ABS,SWAP,ABS
0545 C4E9 C3A3C660C6          FDB USTAR,FROMR,DSETSN
0546 C4EF C05B                FDB SEMIS
0547 C4F1 81AA                FCB $81,'*+$80
0548 C4F3 C4D4                FDB MSTAR-5
0549 C4F5 C04EC4D9C6  STAR    FDB DOCOL,MSTAR,DROP
0550 C4FB C05B                FDB SEMIS
0551 C4FD 824DAF              FCB $82,'M,'/+$80
0552 C500 C4F1                FDB STAR-4
0553 C502 C04EC676C6  MSLASH  FDB DOCOL,OVER,TOR,TOR
0554 C50A C5AFC66DC5          FDB DABS,R,ABS,USLASH
0555 C512 C660C66DC4          FDB FROMR,R,XOR,SETSN
0556 C51A C692C660C5          FDB SWAP,FROMR,SETSN
0557 C520 C692C05B            FDB SWAP,SEMIS
0558 C524 842F4D4FC4          FCB $84,'/,'M,'O,'D+$80
0559 C529 C4FD                FDB MSLASH-5
0560 C52B C04EC652C5  SLMOD   FDB DOCOL,TOR,STOD,FROMR
0561 C533 C502C05B            FDB MSLASH,SEMIS
0562 C537 81AF                FCB $81,'/+$80
0563 C539 C524                FDB SLMOD-7
0564 C53B C04EC52BC6  SLASH   FDB DOCOL,SLMOD,SWAP
0565 C541 C684C05B            FDB DROP,SEMIS
0566 C545 834D4FC4            FCB $83,'M,'O,'D+$80
0567 C549 C537                FDB SLASH-4
0568 C54B C04EC52BC6  MOD     FDB DOCOL,SLMOD,DROP
0569 C551 C05B                FDB SEMIS
```

```
570 C553 852A2F4D4F          FCB $85,'*,'/,'M,'O
571 C558 C4                  FCB 'D+$80
572 C559 C545                FDB MOD-6
573 C55B C04EC652C4  SSMOD   FDB DOCOL,TOR,MSTAR
574 C561 C660C502C0          FDB FROMR,MSLASH,SEMIS
575 C567 822AAF              FCB $82,'*,'/+$80
576 C56A C553                FDB SSMOD-8
577 C56C C04EC55BC6  SSLASH  FDB DOCOL,SSMOD,SWAP
578 C572 C684C05B            FDB DROP,SEMIS
579 C576 854D2F4D4F          FCB $85,'M,'/,'M,'O
580 C57B C4                  FCB 'D+$80
581 C57C C567                FDB SSLASH-5
582 C57E C04EC652C7  MSMOD   FDB DOCOL,TOR,ZERO,R
583 C586 C3D9C660C6          FDB USLASH,FROMR,SWAP
584 C58C C652C3D9C6          FDB TOR,USLASH,FROMR
585 C592 C05B                FDB SEMIS
586 C594 834142D3            FCB $83,'A,'B,'S+$80
587 C598 C576                FDB MSMOD-8
588 C59A C04EC6A3C6  ABS     FDB DOCOL,DUP,ZLESS,ZBRAN
589 C5A2 0004C484            FDB ABS2-*,MINUS
590 C5A6 C05B        ABS2    FDB SEMIS
591 C5A8 84444142D3          FCB $84,'D,'A,'B,'S+$80
592 C5AD C594                FDB ABS-6
593 C5AF C04EC6A3C6  DABS    FDB DOCOL,DUP,ZLESS,ZBRAN
594 C5B7 0004C49D            FDB DABS2-*,DMINUS
595 C5BB C05B        DABS2   FDB SEMIS
596 C5BD 81BC                FCB $81,'<+$80
597 C5BF C5A8                FDB DABS-7
598 C5C1 C5C3        LESS    FDB *+2
599 C5C3 3706                PULU D
600 C5C5 A1C4                CMPA 0,U
601 C5C7 2E09                BGT LESST
602 C5C9 2604                BNE LESSF
603 C5CB E141                CMPB 1,U
604 C5CD 2203                BHI LESST
605 C5CF 5F          LESSF   CLRB
606 C5D0 2002                BRA LESSX
607 C5D2 C601        LESST   LDB #1
608 C5D4 4F          LESSX   CLRA
609 C5D5 16FE59              LBRA PUTD
610 C5D8 84532D3EC4          FCB $84,'S,'-,'>,'D+$80
611 C5DD C5BD                FDB LESS-4
612 C5DF C5E1        STOD    FDB *+2
613 C5E1 CC0000              LDD #0
614 C5E4 6DC4                TST ,U
615 C5E6 2A02                BPL STOD2
616 C5E8 43                  COMA
617 C5E9 53                  COMB
618 C5EA EDC3        STOD2   STD ,--U
619 C5EC 16FA63              LBRA NEXT
620 C5EF 822BAD              FCB $82,'+,'-+$80
621 C5F2 C5D8                FDB STOD-7
```

```
622 C5F4 C04EC62AC1  SETSN  FDB DOCOL,ZLESS,ZBRAN
623 C5FA 0004C484           FDB SETSN2-*,MINUS
624 C5FE C05B        SETSN2 FDB SEMIS
625 C600 83442BAD           FCB $83,'D,'+,'-+$80
626 C604 C5EF               FDB SETSN-5
627 C606 C04EC62AC1  DSETSN FDB DOCOL,ZLESS,ZBRAN
628 C60C 0004C49D           FDB DSETS2-*,DMINUS
629 C610 C05B        DSETS2 FDB SEMIS
630 C612 8230BD             FCB $82,'0,'=+$80
631 C615 C600               FDB DSETSN-6
632 C617 C619        ZEQU   FDB *+2
633 C619 4F                 CLRA
634 C61A 5F                 CLRB
635 C61B AEC4               LDX ,U
636 C61D 2601               BNE ZEQU2
637 C61F 5C                 INCB
638 C620 EDC4        ZEQU2  STD ,U
639 C622 16FA2D             LBRA NEXT
640 C625 8230BC             FCB $82,'0,'<+$80
641 C628 C612               FDB ZEQU-5
642 C62A C62C        ZLESS  FDB *+2
643 C62C 8680               LDA #$80
644 C62E A4C4               ANDA ,U
645 C630 2706               BEQ ZLESS2
646 C632 4F                 CLRA
647 C633 C601               LDB #1
648 C635 16FDF9             LBRA PUTD
649 C638 5F          ZLESS2 CLRB
650 C639 16FDF5             LBRA PUTD
651 C63C 854C454156         FCB $85,'L,'E,'A,'V
652 C641 C5                 FCB 'E+$80
653 C642 C625               FDB ZLESS-5
654 C644 C646        LEAVE  FDB *+2
655 C646 ECE4               LDD ,S
656 C648 ED62               STD 2,S
657 C64A 16FA05             LBRA NEXT
658 C64D 823ED2             FCB $82,'>,'R+$80
659 C650 C63C               FDB LEAVE-8
660 C652 C654        TOR    FDB *+2
661 C654 3706               PULU D
662 C656 3406               PSHS D
663 C658 16F9F7             LBRA NEXT
664 C65B 8252BE             FCB $82,'R,'>+$80
665 C65E C64D               FDB TOR-5
666 C660 C662        FROMR  FDB *+2
667 C662 3506               PULS D
668 C664 3606               PSHU D
669 C666 16F9E9             LBRA NEXT
670 C669 81D2               FCB $81,'R+$80
671 C66B C65B               FDB FROMR-5
672 C66D C24D        R      FDB I+2
673 C66F 844F5645D2         FCB $84,'O,'V,'E,'R+$80
```

```
0674 C674 C669                FDB R-4
0675 C676 C678          OVER  FDB *+2
0676 C678 EC42                LDD 2,U
0677 C67A 16F9B5              LBRA PUSHD
0678 C67D 8444524FD0          FCB $84,'D,'R,'O,'P+$80
0679 C682 C66F                FDB OVER-7
0680 C684 C686          DROP  FDB *+2
0681 C686 3342                LEAU 2,U
0682 C688 16F9C7              LBRA NEXT
0683 C68B 84535741D0          FCB $84,'S,'W,'A,'P+$80
0684 C690 C67D                FDB DROP-7
0685 C692 C694          SWAP  FDB *+2
0686 C694 3716                PULU D,X
0687 C696 1E01                EXG D,X
0688 C698 3616                PSHU D,X
0689 C69A 16F9B5              LBRA NEXT
0690 C69D 834455D0            FCB $83,'D,'U,'P+$80
0691 C6A1 C68B                FDB SWAP-7
0692 C6A3 C6A5          DUP   FDB *+2
0693 C6A5 ECC4                LDD ,U
0694 C6A7 16F988              LBRA PUSHD
0695 C6AA 822BA1              FCB $82,'+,'!+$80
0696 C6AD C69D                FDB DUP-6
0697 C6AF C6B1          PSTORE FDB *+2
0698 C6B1 AEC1                LDX ,U++
0699 C6B3 ECC1                LDD ,U++
0700 C6B5 E384                ADDD ,X
0701 C6B7 ED84                STD ,X
0702 C6B9 16F996              LBRA NEXT
0703 C6BC 81C0                FCB $81,'@+$80
0704 C6BE C6AA                FDB PSTORE-5
0705 C6C0 C6C2          AT    FDB *+2
0706 C6C2 ECD4                LDD [,U]
0707 C6C4 16FD6A              LBRA PUTD
0708 C6C7 8243C0              FCB $82,'C,'@+$80
0709 C6CA C6BC                FDB AT-4
0710 C6CC C6CE          CAT   FDB *+2
0711 C6CE E6D4                LDB [,U]
0712 C6D0 4F                  CLRA
0713 C6D1 16FD5D              LBRA PUTD
0714 C6D4 81A1                FCB $81,'!+$80
0715 C6D6 C6C7                FDB CAT-5
0716 C6D8 C6DA          STORE FDB *+2
0717 C6DA 3710                PULU X
0718 C6DC 3706                PULU D
0719 C6DE ED84                STD ,X
0720 C6E0 16F96F              LBRA NEXT
0721 C6E3 8243A1              FCB $82,'C,'!+$80
0722 C6E6 C6D4                FDB STORE-4
0723 C6E8 C6EA          CSTORE FDB *+2
0724 C6EA 3710                PULU X
0725 C6EC 3706                PULU D
```

```
0726 C6EE E784                STB ,X
0727 C6F0 16F95F              LBRA NEXT
0728 C6F3 873C425549          FCB $87,'<,'B,'U,'I,'L,'D
0729 C6FA D3                  FCB 'S+$80
0730 C6FB C6E3                FDB CSTORE-5
0731 C6FD C04EC784C7  BUILDS  FDB DOCOL,ZERO,CON,SEMIS
0732 C705 85444F4553          FCB $85,'D,'O,'E,'S
0733 C70A BE                  FCB '>+$80
0734 C70B C6F3                FDB BUILDS-$0A
0735 C70D C04EC660C9  DOES    FDB DOCOL,FROMR,LATEST
0736 C713 CA0EC6D8CB          FDB PFA,STORE,PSCODE
0737 C719 3420        DODOES  PSHS Y
0738 C71B 10AE02              LDY 2,X
0739 C71E 3004                LEAX 4,X
0740 C720 3610                PSHU X
0741 C722 16F92D              LBRA NEXT
0742 C725 86544F4747          FCB $86,'T,'O,'G,'G,'L
0743 C72B C5                  FCB 'E+$80
0744 C72C C705                FDB DOES-8
0745 C72E C04EC676C6  TOGGLE  FDB DOCOL,OVER,CAT,XOR
0746 C736 C692C6E8C0          FDB SWAP,CSTORE,SEMIS
0747 C73C C1BB                FCB $C1,';+$80
0748 C73E C725                FDB TOGGLE-9
0749 C740 C04ECA92CA  SEMI    FDB DOCOL,QCSP,COMPIL
0750 C746 C05BCB02            FDB SEMIS,SMUDGE
0751 C74A CADFC05B            FDB LBRAK,SEMIS
0752 C74E 88                  FCB $88
0753 C74F 434F4E5354          FCC /CONSTAN/
0754 C756 D4                  FCB 'T+$80
0755 C757 C73C                FDB SEMI-4
0756 C759 C04ECF3BCB  CON     FDB DOCOL,CREATE,SMUDGE
0757 C75F C906CB40            FDB COMMA,PSCODE
0758 C763 EC02        DOCON   LDD 2,X
0759 C765 16F8CA              LBRA PUSHD
0760 C768 88                  FCB $88
0761 C769 5641524941          FCC /VARIABL/
0762 C770 C5                  FCB 'E+$80
0763 C771 C74E                FDB CON-$0B
0764 C773 C04EC759CB  VAR     FDB DOCOL,CON,PSCODE
0765 C779 3002        DOVAR   LEAX 2,X
0766 C77B 3610                PSHU X
0767 C77D 16F8D2              LBRA NEXT
0768 C780 81B0                FCB $81,'0+$80
0769 C782 C768                FDB VAR-$0B
0770 C784 C7630000    ZERO    FDB DOCON,0
0771 C788 81B1                FCB $81,'1+$80
0772 C78A C780                FDB ZERO-4
0773 C78C C7630001    ONE     FDB DOCON,1
0774 C790 81B2                FCB $81,'2+$80
0775 C792 C788                FDB ONE-4
0776 C794 C7630002    TWO     FDB DOCON,2
0777 C798 81B3                FCB $81,'3+$80
```

```
0778 C79A C790                FDB TWO-4
0779 C79C C7630003    THREE   FDB DOCON,3
0780 C7A0 8242CC              FCB $82,'B,'L+$80
0781 C7A3 C798                FDB THREE-4
0782 C7A5 C7630020    BL      FDB DOCON,$20
0783 C7A9 8546495253          FCB $85,'F,'I,'R,'S
0784 C7AE D4                  FCB 'T+$80
0785 C7AF C7A0                FDB BL-5
0786 C7B1 C76309A0    FIRST   FDB DOCON,VIRBGN
0787 C7B5 854C494D49          FCB $85,'L,'I,'M,'I
0788 C7BA D4                  FCB 'T+$80
0789 C7BB C7A9                FDB FIRST-8
0790 C7BD C04EC809C6  LIMIT   FDB DOCOL,BMAX,AT,LIT
0791 C7C5 0404C4F5C7          FDB 1028,STAR,FIRST
0792 C7CB C459C05B            FDB PLUS,SEMIS
0793 C7CF 84555345D2          FCB $84,'U,'S,'E,'R+$80
0794 C7D4 C7B5                FDB LIMIT-8
0795 C7D6 C04EC759CB  USER    FDB DOCOL,CON,PSCODE
0796 C7DC EC02        DOUSER  LDD 2,X
0797 C7DE F306AA              ADDD UP
0798 C7E1 16F84E              LBRA PUSHD
0799 C7E4 872B4F5249          FCB $87,'+,'O,'R,'I,'G,'I
0800 C7EB CE                  FCB 'N+$80
0801 C7EC C7CF                FDB USER-7
0802 C7EE C04EC1CAC0  PORIG   FDB DOCOL,LIT,PRGBGN
0803 C7F4 C459C05B            FDB PLUS,SEMIS
0804 C7F8 835449C2            FCB $83,'T,'I,'B+$80
0805 C7FC C7E4                FDB PORIG-$0A
0806 C7FE C7DC0018    TIB     FDB DOUSER,$18
0807 C802 84424D41D8          FCB $84,'B,'M,'A,'X+$80
0808 C807 C7F8                FDB TIB-6
0809 C809 C7DC0012    BMAX    FDB DOUSER,$12
0810 C80D 8557494454          FCB $85,'W,'I,'D,'T
0811 C812 C8                  FCB 'H+$80
0812 C813 C802                FDB BMAX-7
0813 C815 C7DC0020    WIDTH   FDB DOUSER,$20
0814 C819 87                  FCB $87
0815 C81A 5741524E49          FCC /WARNIN/
0816 C820 C7                  FCB 'G+$80
0817 C821 C80D                FDB WIDTH-8
0818 C823 C7DC0024    WARN    FDB DOUSER,$24
0819 C827 8546454E43          FCB $85,'F,'E,'N,'C
0820 C82C C5                  FCB 'E+$80
0821 C82D C819                FDB WARN-$0A
0822 C82F C7DC0008    FENCE   FDB DOUSER,8
0823 C833 8244D0              FCB $82,'D,'P+$80
0824 C836 C827                FDB FENCE-8
0825 C838 C7DC000A    DP      FDB DOUSER,$0A
0826 C83C 88                  FCB $88
0827 C83D 564F432D4C          FCC /VOC-LIN/
0828 C844 CB                  FCB 'K+$80
0829 C845 C833                FDB DP-5
```

```
0830 C847 C7DC000C     VOCLIN FDB DOUSER,$0C
0831 C84B 83424CCB            FCB $83,'B,'L,'K+$80
0832 C84F C83C                FDB VOCLIN-$0B
0833 C851 C7DC0026     BLK    FDB DOUSER,$26
0834 C855 8249CE              FCB $82,'I,'N+$80
0835 C858 C84B                FDB BLK-6
0836 C85A C7DC0028     IN     FDB DOUSER,$28
0837 C85E 834F55D4            FCB $83,'O,'U,'T+$80
0838 C862 C855                FDB IN-5
0839 C864 C7DC002A     OUT    FDB DOUSER,$2A
0840 C868 835343D2            FCB $83,'S,'C,'R+$80
0841 C86C C85E                FDB OUT-6
0842 C86E C7DC002C     SCR    FDB DOUSER,$2C
0843 C872 864F464653          FCB $86,'O,'F,'F,'S,'E
0844 C878 D4                  FCB 'T+$80
0845 C879 C868                FDB SCR-6
0846 C87B C7DC002E     OFSET  FDB DOUSER,$2E
0847 C87F 87434F4E54          FCB $87,'C,'O,'N,'T,'E,'X
0848 C886 D4                  FCB 'T+$80
0849 C887 C872                FDB OFSET-9
0850 C889 C7DC0030     CONTXT FDB DOUSER,$30
0851 C88D 8743555252          FCB $87,'C,'U,'R,'R,'E,'N
0852 C894 D4                  FCB 'T+$80
0853 C895 C87F                FDB CONTXT-$0A
0854 C897 C7DC0032     CURENT FDB DOUSER,$32
0855 C89B 8553544154          FCB $85,'S,'T,'A,'T
0856 C8A0 C5                  FCB 'E+$80
0857 C8A1 C88D                FDB CURENT-$0A
0858 C8A3 C7DC0034     STATE  FDB DOUSER,$34
0859 C8A7 84424153C5          FCB $84,'B,'A,'S,'E+$80
0860 C8AC C89B                FDB STATE-8
0861 C8AE C7DC0036     BASE   FDB DOUSER,$36
0862 C8B2 834450CC            FCB $83,'D,'P,'L+$80
0863 C8B6 C8A7                FDB BASE-7
0864 C8B8 C7DC0038     DPL    FDB DOUSER,$38
0865 C8BC 83464CC4            FCB $83,'F,'L,'D+$80
0866 C8C0 C8B2                FDB DPL-6
0867 C8C2 C7DC003A     FLD    FDB DOUSER,$3A
0868 C8C6 834353D0            FCB $83,'C,'S,'P+$80
0869 C8CA C8BC                FDB FLD-6
0870 C8CC C7DC003C     CSP    FDB DOUSER,$3C
0871 C8D0 8252A3              FCB $82,'R,'#+$80
0872 C8D3 C8C6                FDB CSP-6
0873 C8D5 C7DC003E     RNUM   FDB DOUSER,$3E
0874 C8D9 83484CC4            FCB $83,'H,'L,'D+$80
0875 C8DD C8D0                FDB RNUM-5
0876 C8DF C7DC0040     HLD    FDB DOUSER,$40
0877 C8E3 8448455 2C5         FCB $84,'H,'E,'R,'E+$80
0878 C8E8 C8D9                FDB HLD-6
0879 C8EA C04EC838C6   HERE   FDB DOCOL,DP,AT,SEMIS
0880 C8F2 85414C4C4F          FCB $85,'A,'L,'L,'O
0881 C8F7 D4                  FCB 'T+$80
```

```
0882 C8F8 C8E3                FDB HERE-7
0883 C8FA C04EC838C6  ALLOT   FDB DOCOL,DP,PSTORE,SEMIS
0884 C902 81AC                FCB $81,',+$80
0885 C904 C8F2                FDB ALLOT-8
0886 C906 C04EC8EAC6  COMMA   FDB DOCOL,HERE,STORE
0887 C90C C794C8FAC0          FDB TWO,ALLOT,SEMIS
0888 C912 8243AC              FCB $82,'C,',+$80
0889 C915 C902                FDB COMMA-4
0890 C917 C04EC8EAC6  CCOMM   FDB DOCOL,HERE,CSTORE
0891 C91D C78CC8FAC0          FDB ONE,ALLOT,SEMIS
0892 C923 81AD                FCB $81,'-+$80
0893 C925 C912                FDB CCOMM-5
0894 C927 C04EC484    SUB     FDB DOCOL,MINUS
0895 C92B C459C05B            FDB PLUS,SEMIS
0896 C92F 81BD                FCB $81,'=+$80
0897 C931 C923                FDB SUB-4
0898 C933 C04EC927C6  EQUAL   FDB DOCOL,SUB,ZEQU,SEMIS
0899 C93B 81BE                FCB $81,'>+$80
0900 C93D C92F                FDB EQUAL-4
0901 C93F C04EC692C5  GREAT   FDB DOCOL,SWAP,LESS,SEMIS
0902 C947 8553504143          FCB $85,'S,'P,'A,'C
0903 C94C C5                  FCB 'E+$80
0904 C94D C93B                FDB GREAT-4
0905 C94F C04EC7A5C0  SPACE   FDB DOCOL,BL,EMIT,SEMIS
0906 C957 834D49CE            FCB $83,'M,'I,'N+$80
0907 C95B C947                FDB SPACE-8
0908 C95D C04EC676C6  MIN     FDB DOCOL,OVER,OVER
0909 C963 C93FC1E6            FDB GREAT,ZBRAN
0910 C967 0004C692            FDB MIN2-*,SWAP
0911 C96B C684C05B    MIN2    FDB DROP,SEMIS
0912 C96F 834D41D8            FCB $83,'M,'A,'X+$80
0913 C973 C957                FDB MIN-6
0914 C975 C04EC676C6  MAX     FDB DOCOL,OVER,OVER,LESS
0915 C97D C1E6                FDB ZBRAN
0916 C97F 0004C692            FDB MAX2-*,SWAP
0917 C983 C684C05B    MAX2    FDB DROP,SEMIS
0918 C987 842D4455D0          FCB $84,'-,'D,'U,'P+$80
0919 C98C C96F                FDB MAX-6
0920 C98E C04EC6A3C1  DDUP    FDB DOCOL,DUP,ZBRAN
0921 C994 0004C6A3            FDB DDUP2-*,DUP
0922 C998 C05B        DDUP2   FDB SEMIS
0923 C99A 88                  FCB $88
0924 C99B 5452415645          FCC /TRAVERS/
0925 C9A2 C5                  FCB 'E+$80
0926 C9A3 C987                FDB DDUP-7
0927 C9A5 C04EC692    TRAV    FDB DOCOL,SWAP
0928 C9A9 C676C459C1  TRAV2   FDB OVER,PLUS,LIT,$7F
0929 C9B1 C676C6CCC5          FDB OVER,CAT,LESS,ZBRAN
0930 C9B9 FFF0                FDB TRAV2-*
0931 C9BB C692C684C0          FDB SWAP,DROP,SEMIS
0932 C9C1 86                  FCB $86
0933 C9C2 4C41544553          FCC /LATES/
```

```
0934 C9C7 D4                    FCB 'T+$80
0935 C9C8 C99A                  FDB TRAV-$0B
0936 C9CA C04EC897C6  LATEST    FDB DOCOL,CURENT,AT
0937 C9D0 C6C0C05B              FDB AT,SEMIS
0938 C9D4 834C46C1              FCB $83,'L,'F,'A+$80
0939 C9D8 C9C1                  FDB LATEST-9
0940 C9DA C04EC1CA00  LFA       FDB DOCOL,LIT,4
0941 C9E0 C927C05B              FDB SUB,SEMIS
0942 C9E4 834346C1              FCB $83,'C,'F,'A+$80
0943 C9E8 C9D4                  FDB LFA-6
0944 C9EA C04EC794C9  CFA       FDB DOCOL,TWO,SUB,SEMIS
0945 C9F2 834E46C1              FCB $83,'N,'F,'A+$80
0946 C9F6 C9E4                  FDB CFA-6
0947 C9F8 C04EC1CA00  NFA       FDB DOCOL,LIT,5
0948 C9FE C927C78CC4            FDB SUB,ONE,MINUS,TRAV
0949 CA06 C05B                  FDB SEMIS
0950 CA08 835046C1              FCB $83,'P,'F,'A+$80
0951 CA0C C9F2                  FDB NFA-6
0952 CA0E C04EC78CC9  PFA       FDB DOCOL,ONE,TRAV,LIT,5
0953 CA18 C459C05B              FDB PLUS,SEMIS
0954 CA1C 84214353D0            FCB $84,'!,'C,'S,'P+$80
0955 CA21 CA08                  FDB PFA-6
0956 CA23 C04EC19EC8  SCSP      FDB DOCOL,SPAT,CSP,STORE
0957 CA2B C05B                  FDB SEMIS
0958 CA2D 86                    FCB $86
0959 CA2E 3F4552524F            FCC /?ERRO/
0960 CA33 D2                    FCB 'R+$80
0961 CA34 CA1C                  FDB SCSP-7
0962 CA36 C04EC692C1  QERR      FDB DOCOL,SWAP,ZBRAN
0963 CA3C 0008CEBBC1            FDB QERR2-*,ERROR,BRAN
0964 CA42 0004                  FDB QERR3-*
0965 CA44 C684        QERR2     FDB DROP
0966 CA46 C05B        QERR3     FDB SEMIS
0967 CA48 85                    FCB $85
0968 CA49 3F434F4D              FCC /?COM/
0969 CA4D D0                    FCB 'P+$80
0970 CA4E CA2D                  FDB QERR-9
0971 CA50 C04EC8A3C6  QCOMP     FDB DOCOL,STATE,AT,ZEQU
0972 CA58 C1CA0011CA            FDB LIT,$11,QERR,SEMIS
0973 CA60 85                    FCB $85
0974 CA61 3F455845              FCC /?EXE/
0975 CA65 C3                    FCB 'C+$80
0976 CA66 CA48                  FDB QCOMP-8
0977 CA68 C04EC8A3C6  QEXEC     FDB DOCOL,STATE,AT,LIT
0978 CA70 0012CA36C0            FDB $12,QERR,SEMIS
0979 CA76 86                    FCB $86
0980 CA77 3F50414952            FCC /?PAIR/
0981 CA7C D3                    FCB 'S+$80
0982 CA7D CA60                  FDB QEXEC-8
0983 CA7F C04EC927C1  QPAIRS    FDB DOCOL,SUB,LIT,$13
0984 CA87 CA36C05B              FDB QERR,SEMIS
0985 CA8B 843F4353D0            FCB $84,'?,'C,'S,'P+$80
```

```
0986 CA90 CA76                FDB QPAIRS-9
0987 CA92 C04EC19EC8  QCSP    FDB DOCOL,SPAT,CSP,AT,SUB
0988 CA9C C1CA0014            FDB LIT,$14
0989 CAA0 CA36C05B            FDB QERR,SEMIS
0990 CAA4 88                  FCB $88
0991 CAA5 3F4C4F4144          FCC /?LOADIN/
0992 CAAC C7                  FCB 'G+$80
0993 CAAD CA8B                FDB QCSP-7
0994 CAAF C04EC851C6  QLOAD   FDB DOCOL,BLK,AT,ZEQU
0995 CAB7 C1CA0016            FDB LIT,$16
0996 CABB CA36C05B            FDB QERR,SEMIS
0997 CABF 87                  FCB $87
0998 CAC0 434F4D5049          FCC /COMPIL/
0999 CAC6 C5                  FCB 'E+$80
1000 CAC7 CAA4                FDB QLOAD-$0B
1001 CAC9 C04ECA50C6  COMPIL  FDB DOCOL,QCOMP,FROMR
1002 CACF C6A3C4CAC6          FDB DUP,TWOP,TOR,AT
1003 CAD7 C906C05B            FDB COMMA,SEMIS
1004 CADB C1DB                FCB $C1,'[+$80
1005 CADD CABF                FDB COMPIL-$0A
1006 CADF C04EC784C8  LBRAK   FDB DOCOL,ZERO,STATE
1007 CAE5 C6D8C05B            FDB STORE,SEMIS
1008 CAE9 81DD                FCB $81,']+$80
1009 CAEB CADB                FDB LBRAK-4
1010 CAED C04EC1CA00  RBRAK   FDB DOCOL,LIT,$C0
1011 CAF3 C8A3C6D8C0          FDB STATE,STORE,SEMIS
1012 CAF9 86                  FCB $86
1013 CAFA 534D554447          FCC /SMUDG/
1014 CAFF C5                  FCB 'E+$80
1015 CB00 CAE9                FDB RBRAK-4
1016 CB02 C04EC9CAC1  SMUDGE  FDB DOCOL,LATEST,LIT,$20
1017 CB0A C72EC05B            FDB TOGGLE,SEMIS
1018 CB0E 834845D8            FCB $83,'H,'E,'X+$80
1019 CB12 CAF9                FDB SMUDGE-9
1020 CB14 C04EC1CA00  HEX     FDB DOCOL,LIT,16
1021 CB1A C8AEC6D8C0          FDB BASE,STORE,SEMIS
1022 CB20 87                  FCB $87
1023 CB21 444543494D          FCC /DECIMA/
1024 CB27 CC                  FCB 'L+$80
1025 CB28 CB0E                FDB HEX-6
1026 CB2A C04EC1CA00  DEC     FDB DOCOL,LIT,10
1027 CB30 C8AEC6D8C0          FDB BASE,STORE,SEMIS
1028 CB36 87                  FCB $87
1029 CB37 283B434F44          FCC /(;CODE/
1030 CB3D A9                  FCB ')+$80
1031 CB3E CB20                FDB DEC-$0A
1032 CB40 C04EC660C9  PSCODE  FDB DOCOL,FROMR,LATEST
1033 CB46 CA0EC9EAC6          FDB PFA,CFA,STORE,SEMIS
1034 CB4E C5                  FCB $C5
1035 CB4F 3B434F44            FCC /;COD/
1036 CB53 C5                  FCB 'E+$80
1037 CB54 CB36                FDB PSCODE-$0A
```

```
1038 CB56 C04ECA92CA  SEMIC  FDB DOCOL,QCSP,COMPIL
1039 CB5C CB40CB02CA         FDB PSCODE,SMUDGE,LBRAK
1040 CB62 CC30               FDB QSTACK
1041 CB64 065CC05B           FDB ASSEM-RAM+BRAM,SEMIS
1042 CB68 85                 FCB $85
1043 CB69 434F554E           FCC /COUN/
1044 CB6D D4                 FCB 'T+$80
1045 CB6E CB4E               FDB SEMIC-8
1046 CB70 C04EC6A3C4  COUNT  FDB DOCOL,DUP,ONEP,SWAP
1047 CB78 C6CCC05B           FDB CAT,SEMIS
1048 CB7C 84545950C5         FCB $84,'T,'Y,'P,'E+$80
1049 CB81 CB68               FDB COUNT-8
1050 CB83 C04EC98EC1  TYPE   FDB DOCOL,DDUP,ZBRAN
1051 CB89 0018C676C4         FDB TYPE3-*,OVER,PLUS
1052 CB8F C692C23C           FDB SWAP,XDO
1053 CB93 C24BC6CCC0  TYPE2  FDB I,CAT,EMIT,XLOOP
1054 CB9B FFF8C1DA           FDB TYPE2-*,BRAN
1055 CB9F 0004               FDB TYPE4-*
1056 CBA1 C684       TYPE3   FDB DROP
1057 CBA3 C05B       TYPE4   FDB SEMIS
1058 CBA5 89                 FCB $89
1059 CBA6 2D54524149         FCC /-TRAILIN/
1060 CBAE C7                 FCB 'G+$80
1061 CBAF CB7C               FDB TYPE-7
1062 CBB1 C04EC6A3C7  DTRAIL FDB DOCOL,DUP,ZERO,XDO
1063 CBB9 C676C676C4  DTRAL2 FDB OVER,OVER,PLUS,ONE
1064 CBC1 C927C6CCC7         FDB SUB,CAT,BL,SUB,ZBRAN
1065 CBCB 0008C644C1         FDB DTRAL3-*,LEAVE,BRAN
1066 CBD1 0006               FDB DTRAL4-*
1067 CBD3 C78CC927    DTRAL3 FDB ONE,SUB
1068 CBD7 C203        DTRAL4 FDB XLOOP
1069 CBD9 FFE0               FDB DTRAL2-*
1070 CBDB C05B               FDB SEMIS
1071 CBDD C4282E22A9         FCB $C4,'(,'.,'",')+$80
1072 CBE2 CBA5               FDB DTRAIL-$0C
1073 CBE4 C04EC66DCB  PDOTQ  FDB DOCOL,R,COUNT,DUP
1074 CBEC C4BBC660C4         FDB ONEP,FROMR,PLUS,TOR
1075 CBF4 CB83C05B           FDB TYPE,SEMIS
1076 CBF8 C22EA2             FCB $C2,'.,'"+$80
1077 CBFB CBDD               FDB PDOTQ-7
1078 CBFD C04EC1CA00  DOTQ   FDB DOCOL,LIT,$22
1079 CC03 C8A3C6C0C1         FDB STATE,AT,ZBRAN
1080 CC09 0014               FDB DOTQ1-*
1081 CC0B CAC9CBE4CD         FDB COMPIL,PDOTQ,WORD
1082 CC11 C8EAC6CCC4         FDB HERE,CAT,ONEP,ALLOT
1083 CC19 C1DA               FDB BRAN
1084 CC1B 000A               FDB DOTQ2-*
1085 CC1D CD9DC8EACB  DOTQ1  FDB WORD,HERE,COUNT,TYPE
1086 CC25 C05B       DOTQ2   FDB SEMIS
1087 CC27 86                 FCB $86
1088 CC28 3F53544143         FCC /?STAC/
1089 CC2D CB                 FCB 'K+$80
```

```
1090 CC2E CBF8                FDB DOTQ-5
1091 CC30 C04EC19EC1  QSTACK  FDB DOCOL,SPAT,LIT
1092 CC36 001AC7EEC6          FDB SINIT-PRGBGN,PORIG,AT
1093 CC3C C692C5C1C7          FDB SWAP,LESS,ONE,QERR
1094 CC44 C1CA0074C6          FDB LIT,$74,AT  MEMEND
1095 CC4A C8EAC1CA00          FDB HERE,LIT,$80
1096 CC50 C459C5C1C7          FDB PLUS,LESS,TWO,QERR
1097 CC58 C05B                FDB SEMIS
1098 CC5A 83524FD4            FCB $83,'R,'O,'T+$80
1099 CC5E CC27                FDB QSTACK-9
1100 CC60 C04EC652C6  ROT     FDB DOCOL,TOR,SWAP,FROMR
1101 CC68 C692C05B            FDB SWAP,SEMIS
1102 CC6C 86                  FCB $86
1103 CC6D 4558504543          FCC /EXPEC/
1104 CC72 D4                  FCB 'T+$80
1105 CC73 CC5A                FDB ROT-6
1106 CC75 C04EC676C4  EXPECT  FDB DOCOL,OVER,PLUS,OVER
1107 CC7D C23C                FDB XDO
1108 CC7F C0D6C6A3C1  EXPEC2  FDB KEY,DUP,LIT,08
1109 CC87 C933C1E6            FDB EQUAL,ZBRAN
1110 CC8B 0020C684C1          FDB EXPEC3-*,DROP,LIT
1111 CC91 0008C676C2          FDB 08,OVER,I
1112 CC97 C933C6A3C6          FDB EQUAL,DUP,FROMR,TWO
1113 CC9F C927C459C6          FDB SUB,PLUS,TOR,SUB,BRAN
1114 CCA9 0028                FDB EXPEC6-*
1115 CCAB C6A3C1CA00  EXPEC3  FDB DUP,LIT,$0D
1116 CCB1 C933C1E6            FDB EQUAL,ZBRAN
1117 CCB5 000E                FDB EXPEC4-*
1118 CCB7 C644C684C7          FDB LEAVE,DROP,BL,ZERO
1119 CCBF C1DA                FDB BRAN
1120 CCC1 0004                FDB EXPEC5-*
1121 CCC3 C6A3        EXPEC4  FDB DUP
1122 CCC5 C24BC6E8C7  EXPEC5  FDB I,CSTORE,ZERO,I,ONEP
1123 CCCF C6D8                FDB STORE
1124 CCD1 C09DC203    EXPEC6  FDB EMIT,XLOOP
1125 CCD5 FFAA                FDB EXPEC2-*
1126 CCD7 C684C05B            FDB DROP,SEMIS
1127 CCDB 85                  FCB $85
1128 CCDC 51554552            FCC /QUER/
1129 CCE0 D9                  FCB 'Y+$80
1130 CCE1 CC6C                FDB EXPECT-9
1131 CCE3 C04EC7FEC6  QUERY   FDB DOCOL,TIB,AT,LIT,80
1132 CCED CC75C784C8          FDB EXPECT,ZERO,IN
1133 CCF3 C6D8C05B            FDB STORE,SEMIS
1134 CCF7 C180                FCB $C1,$80
1135 CCF9 CCDB                FDB QUERY-8
1136 CCFB C04EC851C6  NULL    FDB DOCOL,BLK,AT,ZBRAN
1137 CD03 0022                FDB NULL2-*
1138 CD05 C78CC851C6          FDB ONE,BLK,PSTORE,ZERO
1139 CD0D C85AC6D8C8          FDB IN,STORE,BLK,AT
1140 CD15 C617C1E6            FDB ZEQU,ZBRAN
1141 CD19 0008                FDB NULL1-*
```

```
1142 CD1B CA68C660C6          FDB QEXEC,FROMR,DROP
1143 CD21 C1DA          NULL1 FDB BRAN
1144 CD23 0006                FDB NULL3-*
1145 CD25 C660C684      NULL2 FDB FROMR,DROP
1146 CD29 C05B          NULL3 FDB SEMIS
1147 CD2B 8446494CCC          FCB $84,'F,'I,'L,'L+$80
1148 CD30 CCF7                FDB NULL-4
1149 CD32 C04EC692C6    FILL  FDB DOCOL,SWAP,TOR,OVER
1150 CD3A C6E8C6A3C4          FDB CSTORE,DUP,ONEP,FROMR
1151 CD42 C78CC927C3          FDB ONE,SUB,CMOVE,SEMIS
1152 CD4A 8545524153          FCB $85,'E,'R,'A,'S
1153 CD4F C5                  FCB 'E+$80
1154 CD50 CD2B                FDB FILL-7
1155 CD52 C04EC784CD    ERASE FDB DOCOL,ZERO,FILL,SEMIS
1156 CD5A 86424C414E          FCB $86,'B,'L,'A,'N,'K
1157 CD60 D3                  FCB 'S+$80
1158 CD61 CD4A                FDB ERASE-8
1159 CD63 C04EC7A5CD    BLANKS FDB DOCOL,BL,FILL,SEMIS
1160 CD6B 84484F4CC4          FCB $84,'H,'O,'L,'D+$80
1161 CD70 CD5A                FDB BLANKS-9
1162 CD72 C04EC1CAFF    HOLD  FDB DOCOL,LIT,-1,HLD
1163 CD7A C6AFC8DFC6          FDB PSTORE,HLD,AT
1164 CD80 C6E8C05B            FDB CSTORE,SEMIS
1165 CD84 835041C4            FCB $83,'P,'A,'D+$80
1166 CD88 CD6B                FDB HOLD-7
1167 CD8A C04EC8EAC1    PAD   FDB DOCOL,HERE,LIT,$44
1168 CD92 C459C05B            FDB PLUS,SEMIS
1169 CD96 84574F52C4          FCB $84,'W,'O,'R,'D+$80
1170 CD9B CD84                FDB PAD-6
1171 CD9D C04EC851C6    WORD  FDB DOCOL,BLK,AT,ZBRAN
1172 CDA5 000CC851C6          FDB WORD2-*,BLK,AT
1173 CDAB 0638                FDB BLOCK-RAM+BRAM
1174 CDAD C1DA                FDB BRAN
1175 CDAF 0006                FDB WORD3-*
1176 CDB1 C7FEC6C0      WORD2 FDB TIB,AT
1177 CDB5 C85AC6C0C4    WORD3 FDB IN,AT,PLUS,SWAP
1178 CDBD C2DDC8EAC1          FDB ENCLOS,HERE,LIT,$34
1179 CDC5 CD63C85AC6          FDB BLANKS,IN,PSTORE,OVER
1180 CDCD C927C652C6          FDB SUB,TOR,R,HERE,CSTORE
1181 CDD7 C459C8EAC4          FDB PLUS,HERE,ONEP,FROMR
1182 CDDF C31AC05B            FDB CMOVE,SEMIS
1183 CDE3 88                  FCB $88
1184 CDE4 284E554D42          FCC /(NUMBER/
1185 CDEB A9                  FCB ')+$80
1186 CDEC CD96                FDB WORD-7
1187 CDEE C04E          PNUMB FDB DOCOL
1188 CDF0 C4BBC6A3C6    PNUMB2 FDB ONEP,DUP,TOR,CAT,BASE
1189 CDFA C6C0C25AC1          FDB AT,DIGIT,ZBRAN
1190 CE00 002CC692C8          FDB PNUMB4-*,SWAP,BASE,AT
1191 CE08 C3A3C684CC          FDB USTAR,DROP,ROT,BASE
1192 CE10 C6C0C3A3C4          FDB AT,USTAR,DPLUS,DPL
1193 CE18 C6C0C4BBC1          FDB AT,ONEP,ZBRAN
```

```
1194 CE1E 0008C78CC8          FDB PNUMB3-*,ONE,DPL
1195 CE24 C6AF                FDB PSTORE
1196 CE26 C660C1DA    PNUMB3  FDB FROMR,BRAN
1197 CE2A FFC6                FDB PNUMB2-*
1198 CE2C C660C05B    PNUMB4  FDB FROMR,SEMIS
1199 CE30 864E554D42          FCB $86,'N,'U,'M,'B,'E
1200 CE36 D2                  FCB 'R+$80
1201 CE37 CDE3                FDB PNUMB-$0B
1202 CE39 C04EC784C7  NUMB    FDB DOCOL,ZERO,ZERO,ROT
1203 CE41 C6A3C4BBC6          FDB DUP,ONEP,CAT,LIT,$2D
1204 CE4B C933C6A3C6          FDB EQUAL,DUP,TOR,PLUS
1205 CE53 C1CAFFFF            FDB LIT,-1
1206 CE57 C8B8C6D8CD  NUMB1   FDB DPL,STORE,PNUMB,DUP
1207 CE5F C6CCC7A5C9          FDB CAT,BL,SUB,ZBRAN
1208 CE67 0016                FDB NUMB2-*
1209 CE69 C6A3C6CCC1          FDB DUP,CAT,LIT,$2E,SUB
1210 CE73 C784CA36C7          FDB ZERO,QERR,ZERO,BRAN
1211 CE7B FFDC                FDB NUMB1-*
1212 CE7D C684C660C1  NUMB2   FDB DROP,FROMR,ZBRAN
1213 CE83 0004C49D            FDB NUMB3-*,DMINUS
1214 CE87 C05B        NUMB3   FDB SEMIS
1215 CE89 852D46494E          FCB $85,'-,'F,'I,'N
1216 CE8E C4                  FCB 'D+$80
1217 CE8F CE30                FDB NUMB-9
1218 CE91 C04EC7A5CD  DFIND   FDB DOCOL,BL,WORD,HERE
1219 CE99 C889C6C0C6          FDB CONTXT,AT,AT,PFIND
1220 CEA1 C6A3C617C1          FDB DUP,ZEQU,ZBRAN
1221 CEA7 000AC684C8          FDB DFIND2-*,DROP,HERE
1222 CEAD C9CAC28D            FDB LATEST,PFIND
1223 CEB1 C05B        DFIND2  FDB SEMIS
1224 CEB3 854552524F          FCB $85,'E,'R,'R,'O
1225 CEB8 D2                  FCB 'R+$80
1226 CEB9 CE89                FDB DFIND-8
1227 CEBB C04EC823C6  ERROR   FDB DOCOL,WARN,AT,ZLESS
1228 CEC3 C1E6                FDB ZBRAN
1229 CEC5 0004                FDB ERROR2-*
1230 CEC7 062A                FDB PABORT-RAM+BRAM
1231 CEC9 C8EACB70CB  ERROR2  FDB HERE,COUNT,TYPE,PDOTQ
1232 CED1 043C2D3F20          FCB 4,'<,'-,'?,32
1233 CED6 C1CA00EEC7          FDB LIT,238,ONE,TONE
1234 CEDE D150C1AD            FDB MESS,SPSTOR
1235 CEE2 D0AEC05B            FDB QUIT,SEMIS
1236 CEE6 834944AE            FCB $83,'I,'D,'.+$80
1237 CEEA CEB3                FDB ERROR-8
1238 CEEC C04ECD8AC1  IDDOT   FDB DOCOL,PAD,LIT,32
1239 CEF4 C1CA005F            FDB LIT,$5F
1240 CEF8 CD32C6A3CA          FDB FILL,DUP,PFA,LFA,OVER
1241 CF02 C927CD8AC6          FDB SUB,PAD,SWAP,CMOVE
1242 CF0A CD8ACB70C1          FDB PAD,COUNT,LIT,31,AND
1243 CF14 C676C676C4          FDB OVER,OVER,PLUS
1244 CF1A C78CC927C1          FDB ONE,SUB,LIT
1245 CF20 007FC676C6          FDB $7F,OVER,CAT,AND,SWAP
```

```
1246 CF2A C6E8CB83            FDB CSTORE,TYPE
1247 CF2E C94FC05B            FDB SPACE,SEMIS
1248 CF32 8643524541          FCB $86,'C,'R,'E,'A,'T
1249 CF38 C5                  FCB 'E+$80
1250 CF39 CEE6                FDB IDDOT-6
1251 CF3B C04ECE91C1  CREATE  FDB DOCOL,DFIND,ZBRAN
1252 CF41 0014C684CB          FDB CREAT2-*,DROP,PDOTQ
1253 CF47 07                  FCB 7
1254 CF48 5245444546          FCC /REDEF: /
1255 CF4F C9F8CEECC9          FDB NFA,IDDOT,SPACE
1256 CF55 C8EAC6A3C6  CREAT2  FDB HERE,DUP,CAT,WIDTH,AT
1257 CF5F C95DC4BBC8          FDB MIN,ONEP,ALLOT,DUP
1258 CF67 C1CA00A0            FDB LIT,$A0
1259 CF6B C72EC8EAC7          FDB TOGGLE,HERE,ONE,SUB
1260 CF73 C1CA0080            FDB LIT,$80
1261 CF77 C72EC9CAC9          FDB TOGGLE,LATEST,COMMA
1262 CF7D C897C6C0C6          FDB CURENT,AT,STORE,HERE
1263 CF85 C4CAC906C0          FDB TWOP,COMMA,SEMIS
1264 CF8B C95B434F4D          FCB $C9,'[,'C,'O,'M,'P,'I
1265 CF92 4C45DD              FCB 'L,'E,']+$80
1266 CF95 CF32                FDB CREATE-9
1267 CF97 C04ECE91C6  BCOMP   FDB DOCOL,DFIND,ZEQU,ZERO
1268 CF9F CA36C684C9          FDB QERR,DROP,CFA,COMMA
1269 CFA7 C05B                FDB SEMIS
1270 CFA9 874C495445          FCB $87,'L,'I,'T,'E,'R,'A
1271 CFB0 CC                  FCB 'L+$80
1272 CFB1 CF8B                FDB BCOMP-$0C
1273 CFB3 C04EC8A3C6  LITER   FDB DOCOL,STATE,AT,ZBRAN
1274 CFBB 0008CAC9C1          FDB LITER2-*,COMPIL,LIT
1275 CFC1 C906                FDB COMMA
1276 CFC3 C05B        LITER2  FDB SEMIS
1277 CFC5 88444C4954          FCB $88,'D,'L,'I,'T,'E,'R
1278 CFCC 41CC                FCB 'A,'L+$80
1279 CFCE CFA9                FDB LITER-10
1280 CFD0 C04EC8A3C6  DLITER  FDB DOCOL,STATE,AT,ZBRAN
1281 CFD8 0008C692CF          FDB DLITE2-*,SWAP,LITER
1282 CFDE CFB3                FDB LITER
1283 CFE0 C05B        DLITE2  FDB SEMIS
1284 CFE2 89494E5445          FCB $89,'I,'N,'T,'E,'R,'P
1285 CFE9 5245D4              FCB 'R,'E,'T+$80
1286 CFEC CFC5                FDB DLITER-$0B
1287 CFEE C04E        INTERP  FDB DOCOL
1288 CFF0 CE91C1E6    INTER2  FDB DFIND,ZBRAN
1289 CFF4 001EC8A3C6          FDB INTER5-*,STATE,AT
1290 CFFA C5C1C1E6            FDB LESS,ZBRAN
1291 CFFE 000AC9EAC9          FDB INTER3-*,CFA,COMMA
1292 D004 C1DA                FDB BRAN
1293 D006 0006                FDB INTER4-*
1294 D008 C9EAC06C    INTER3  FDB CFA,EXEC
1295 D00C CC30C1DA    INTER4  FDB QSTACK,BRAN
1296 D010 001A                FDB INTER7-*
1297 D012 C8EACE39C8  INTER5  FDB HERE,NUMB,DPL,AT,ONEP
```

```
1298 D01C C1E6                FDB ZBRAN
1299 D01E 0008CFD0C1          FDB INTER6-*,DLITER,BRAN
1300 D024 0006                FDB INTER7-*
1301 D026 C684CFB3  INTER6    FDB DROP,LITER
1302 D02A CC30C1DA  INTER7    FDB QSTACK,BRAN
1303 D02E FFC2                FDB INTER2-*
1304 D030 89494D4D45          FCB $89,'I,'M,'M,'E,'D,'I
1305 D037 4154C5              FCB 'A,'T,'E+$80
1306 D03A CFE2                FDB INTERP-$0C
1307 D03C C04EC9CAC1  IMMED   FDB DOCOL,LATEST,LIT,$40
1308 D044 C72EC05B            FDB TOGGLE,SEMIS
1309 D048 8A                  FCB $8A
1310 D049 564F434142          FCC /VOCABULAR/
1311 D052 D9                  FCB 'Y+$80
1312 D053 D030                FDB IMMED-$0C
1313 D055 C04EC6FDC1  VOCAB   FDB DOCOL,BUILDS,LIT
1314 D05B 81A0C906C8          FDB $81A0,COMMA,CURENT,AT
1315 D063 C9EAC906C8          FDB CFA,COMMA,HERE,VOCLIN
1316 D06B C6C0C906C8          FDB AT,COMMA,VOCLIN,STORE
1317 D073 C70D                FDB DOES
1318 D075 C4CAC889C6  DOVOC   FDB TWOP,CONTXT,STORE
1319 D07B C05B                FDB SEMIS
1320 D07D 0000                FDB 0
1321 D07F 8B44454649          FCB $8B,'D,'E,'F,'I,'N,'I
1322 D086 54494F4ED3          FCB 'T,'I,'O,'N,'S+$80
1323 D08B D048                FDB VOCAB-$0D
1324 D08D C04EC889C6  DEFIN   FDB DOCOL,CONTXT,AT
1325 D093 C897C6D8C0          FDB CURENT,STORE,SEMIS
1326 D099 C1A8                FCB $C1,'(+$80
1327 D09B D07F                FDB DEFIN-$0E
1328 D09D C04EC1CA00  PAREN   FDB DOCOL,LIT,$29  ')'
1329 D0A3 CD9DC05B            FDB WORD,SEMIS
1330 D0A7 84515549D4          FCB $84,'Q,'U,'I,'T+$80
1331 D0AC D099                FDB PAREN-4
1332 D0AE C04EC784C8  QUIT    FDB DOCOL,ZERO,BLK,STORE
1333 D0B6 CADF                FDB LBRAK
1334 D0B8 C1BBC112CC  QUIT2   FDB RPSTOR,CR,QUERY
1335 D0BE CFEEC8A3C6          FDB INTERP,STATE,AT,ZEQU
1336 D0C6 C1E6                FDB ZBRAN
1337 D0C8 0008                FDB QUIT3-*
1338 D0CA CBE4                FDB PDOTQ
1339 D0CC 03204F4B            FCB 3,32,'O,'K
1340 D0D0 C1DA        QUIT3   FDB BRAN
1341 D0D2 FFE6                FDB QUIT2-*
1342 D0D4 8541424F52          FCB $85,'A,'B,'O,'R
1343 D0D9 D4                  FCB 'T+$80
1344 D0DA D0A7                FDB QUIT-7
1345 D0DC C04EC1ADCB  ABORT   FDB DOCOL,SPSTOR,DEC
1346 D0E2 C112CBE4            FDB CR,PDOTQ
1347 D0E6 29                  FCB 41
1348 D0E7 434F4C4F52          FCC /COLORFORTH 1.0/
1349 D0F5 0D0A07              FCB 13,10,07
```

```
1350 D0F8 434F505952          FCC /COPYRIGHT 1981 /
1351 D107 544A5A2026          FCC /TJZ & RJT/
1352 D110 C112C784C8          FDB CR,ZERO,IN,STORE,ZERO
1353 D11A C851C6D8            FDB BLK,STORE
1354 D11E 0646                FDB FORTH-RAM+BRAM
1355 D120 D08DD0AE            FDB DEFIN,QUIT
1356 D124 85422F4255          FCB $85,'B,'/,'B,'U
1357 D129 C6                  FCB 'F+$80
1358 D12A D0D4                FDB ABORT-8
1359 D12C C7630400    BBUF    FDB DOCON,1024
1360 D130 85422F5343          FCB $85,'B,'/,'S,'C
1361 D135 D2                  FCB 'R+$80
1362 D136 D124                FDB BBUF-8
1363 D138 C7630001    BSCR    FDB DOCON,1
1364 D13C 83432FCC            FCB $83,'C,'/,'L+$80
1365 D140 D130                FDB BSCR-8
1366 D142 C7630020    CSL     FDB DOCON,32
1367 D146 874D455353          FCB $87,'M,'E,'S,'S,'A
1368 D14C 47C5                FCB 'G,'E+$80
1369 D14E D13C                FDB CSL-6
1370 D150 C04ED32DC0  MESS    FDB DOCOL,DOT,SEMIS
1371 D156 C1A7                FCB $C1,''+$80
1372 D158 D146                FDB MESS-$0A
1373 D15A C04ECE91C6  TICK    FDB DOCOL,DFIND,ZEQU,ZERO
1374 D162 CA36C684CF          FDB QERR,DROP,LITER,SEMIS
1375 D16A 86464F5247          FCB $86,'F,'O,'R,'G,'E
1376 D170 D4                  FCB 'T+$80
1377 D171 D156                FDB TICK-4
1378 D173 C04EC897C6  FORGET  FDB DOCOL,CURENT,AT
1379 D179 C889C6C0C9          FDB CONTXT,AT,SUB,LIT,$18
1380 D183 CA36D15AC6          FDB QERR,TICK,DUP,FENCE
1381 D18B C6C0C5C1C1          FDB AT,LESS,LIT,$15,QERR
1382 D195 C6A3C9F8C8          FDB DUP,NFA,DP,STORE,LFA
1383 D19F C6C0C889C6          FDB AT,CONTXT,AT,STORE
1384 D1A7 C05B                FDB SEMIS
1385 D1A9 84424143CB          FCB $84,'B,'A,'C,'K+$80
1386 D1AE D16A                FDB FORGET-9
1387 D1B0 C04EC8EAC9  BACK    FDB DOCOL,HERE,SUB,COMMA
1388 D1B8 C05B                FDB SEMIS
1389 D1BA C542454749          FCB $C5,'B,'E,'G,'I
1390 D1BF CE                  FCB 'N+$80
1391 D1C0 D1A9                FDB BACK-7
1392 D1C2 C04ECA50C8  BEGIN   FDB DOCOL,QCOMP,HERE
1393 D1C8 C78CC05B            FDB ONE,SEMIS
1394 D1CC C5454E4449          FCB $C5,'E,'N,'D,'I
1395 D1D1 C6                  FCB 'F+$80
1396 D1D2 D1BA                FDB BEGIN-8
1397 D1D4 C04ECA50C7  ENDIF   FDB DOCOL,QCOMP,TWO
1398 D1DA CA7FC8EAC6          FDB QPAIRS,HERE,OVER,SUB
1399 D1E2 C692C6D8C0          FDB SWAP,STORE,SEMIS
1400 D1E8 C244CF              FCB $C2,'D,'O+$80
1401 D1EB D1CC                FDB ENDIF-8
```

```
1402 D1ED C04ECAC9C2  DO      FDB DOCOL,COMPIL,XDO,HERE
1403 D1F5 C79CC05B            FDB THREE,SEMIS
1404 D1F9 C44C4F4FD0          FCB $C4,'L,'O,'O,'P+$80
1405 D1FE D1E8                FDB DO-5
1406 D200 C04EC79CCA  LOOP    FDB DOCOL,THREE,QPAIRS
1407 D206 CAC9C203            FDB COMPIL,XLOOP
1408 D20A D1B0C05B            FDB BACK,SEMIS
1409 D20E C249C6              FCB $C2,'I,'F+$80
1410 D211 D1F9                FDB LOOP-7
1411 D213 C04ECAC9C1  IF      FDB DOCOL,COMPIL,ZBRAN
1412 D219 C8EAC784C9          FDB HERE,ZERO,COMMA
1413 D21F C794C05B            FDB TWO,SEMIS
1414 D223 C4454C53C5          FCB $C4,'E,'L,'S,'E+$80
1415 D228 D20E                FDB IF-5
1416 D22A C04EC794CA  ELSE    FDB DOCOL,TWO,QPAIRS
1417 D230 CAC9C1DAC8          FDB COMPIL,BRAN,HERE,ZERO
1418 D238 C906C692C7          FDB COMMA,SWAP,TWO,ENDIF
1419 D240 C794C05B            FDB TWO,SEMIS
1420 D244 8653504143          FCB $86,'S,'P,'A,'C,'E
1421 D24A D3                  FCB 'S+$80
1422 D24B D223                FDB ELSE-7
1423 D24D C04EC784C9  SPACES  FDB DOCOL,ZERO,MAX,DDUP
1424 D255 C1E6                FDB ZBRAN
1425 D257 000CC784C2          FDB SPACE3-*,ZERO,XDO
1426 D25D C94FC203    SPACE2  FDB SPACE,XLOOP
1427 D261 FFFC                FDB SPACE2-*
1428 D263 C05B        SPACE3  FDB SEMIS
1429 D265 823CA3              FCB $82,'<,'#+$80
1430 D268 D244                FDB SPACES-9
1431 D26A C04ECD8AC8  BDIGS   FDB DOCOL,PAD,HLD
1432 D270 C6D8C05B            FDB STORE,SEMIS
1433 D274 8223BE              FCB $82,'#,'>+$80
1434 D277 D265                FDB BDIGS-5
1435 D279 C04EC684C6  EDIGS   FDB DOCOL,DROP,DROP,HLD
1436 D281 C6C0CD8AC6          FDB AT,PAD,OVER,SUB,SEMIS
1437 D28B 84534947CE          FCB $84,'S,'I,'G,'N+$80
1438 D290 D274                FDB EDIGS-5
1439 D292 C04ECC60C6  SIGN    FDB DOCOL,ROT,ZLESS,ZBRAN
1440 D29A 0008C1CA00          FDB SIGN2-*,LIT,$2D,HOLD
1441 D2A2 C05B        SIGN2   FDB SEMIS
1442 D2A4 81A3                FCB $81,'#+$80
1443 D2A6 D28B                FDB SIGN-7
1444 D2A8 C04EC8AEC6  DIG     FDB DOCOL,BASE,AT,MSMOD
1445 D2B0 CC60C1CA00          FDB ROT,LIT,9,OVER,LESS
1446 D2BA C1E6                FDB ZBRAN
1447 D2BC 0008C1CA00          FDB DIG2-*,LIT,7,PLUS
1448 D2C4 C1CA0030C4  DIG2    FDB LIT,$30,PLUS
1449 D2CA CD72C05B            FDB HOLD,SEMIS
1450 D2CE 8223D3              FCB $82,'#,'S+$80
1451 D2D1 D2A4                FDB DIG-4
1452 D2D3 C04E        DIGS    FDB DOCOL
1453 D2D5 D2A8C676C6  DIGS2   FDB DIG,OVER,OVER,OR
```

```
1454 D2DD C617C1E6           FDB ZEQU,ZBRAN
1455 D2E1 FFF4C05B           FDB DIGS2-*,SEMIS
1456 D2E5 83442ED2           FCB $83,'D,'.,'R+$80
1457 D2E9 D2CE               FDB DIGS-5
1458 D2EB C04EC652C6  DDOTR  FDB DOCOL,TOR,SWAP,OVER
1459 D2F3 C5AFD26AD2         FDB DABS,BDIGS,DIGS,SIGN
1460 D2FB D279C660C6         FDB EDIGS,FROMR,OVER,SUB
1461 D303 D24DCB83C0         FDB SPACES,TYPE,SEMIS
1462 D309 822ED2             FCB $82,'.,'R+$80
1463 D30C D2E5               FDB DDOTR-6
1464 D30E C04EC652C5  DOTR   FDB DOCOL,TOR,STOD,FROMR
1465 D316 D2EBC05B           FDB DDOTR,SEMIS
1466 D31A 8244AE             FCB $82,'D,'.+$80
1467 D31D D309               FDB DOTR-5
1468 D31F C04EC784D2  DDOT   FDB DOCOL,ZERO,DDOTR
1469 D325 C94FC05B           FDB SPACE,SEMIS
1470 D329 81AE               FCB $81,'.+$80
1471 D32B D31A               FDB DDOT-5
1472 D32D C04EC5DFD3  DOT    FDB DOCOL,STOD,DDOT,SEMIS
1473 D335 85564C4953         FCB $85,'V,'L,'I,'S
1474 D33A D4                 FCB 'T+$80
1475 D33B D329               FDB DOT-4
1476 D33D C04EC889C6  VLIST  FDB DOCOL,CONTXT,AT,AT
1477 D345 C6A3C6CCC1 VLIST1 FDB DUP,CAT,LIT,31,AND
1478 D34F C864C6C0C4         FDB OUT,AT,PLUS,CSL,THREE
1479 D359 C927C93FC1         FDB SUB,GREAT,ZBRAN
1480 D35F 0004C112           FDB VLIST2-*,CR
1481 D363 C6A3CEECC9 VLIST2 FDB DUP,IDDOT,SPACE
1482 D369 CA0EC9DAC6         FDB PFA,LFA,AT,DUP,ZEQU
1483 D373 C0FDC43BC1         FDB QTERM,OR,ZBRAN
1484 D379 FFCCC684C0         FDB VLIST1-*,DROP,SEMIS
1485 D37F 8728424C4F         FCB $87,'(,'B,'L,'O,'C
1486 D385 4BA9               FCB 'K,')+$80
1487 D387 D335               FDB VLIST-8
1488 D389 C04EC652C6  PBLOCK FDB DOCOL,TOR,R,ONE,LESS
1489 D393 C66DC809C6         FDB R,BMAX,AT,GREAT
1490 D39B C43BC1CA00         FDB OR,LIT,8,QERR,FROMR
1491 D3A5 C78CC927D1         FDB ONE,SUB,BBUF,LIT,4
1492 D3AF C459C4F5C7         FDB PLUS,STAR,FIRST
1493 D3B5 C459C4CAC0         FDB PLUS,TWOP,SEMIS
1494 D3BB 844C4F41C4         FCB $84,'L,'O,'A,'D+$80
1495 D3C0 D37F               FDB PBLOCK-10
1496 D3C2 C04EC851C6  LOAD   FDB DOCOL,BLK,AT,TOR,IN
1497 D3CC C6C0C652C7         FDB AT,TOR,ZERO,IN,STORE
1498 D3D6 C851C6D8CF         FDB BLK,STORE,INTERP
1499 D3DC C660C85AC6         FDB FROMR,IN,STORE,FROMR
1500 D3E4 C851C6D8C0         FDB BLK,STORE,SEMIS
1501 D3EA 86284C494E         FCB $86,'(,'L,'I,'N,'E
1502 D3F0 A9                 FCB ')+$80
1503 D3F1 D3BB               FDB LOAD-7
1504 D3F3 C04EC652D1  PLINE  FDB DOCOL,TOR,CSL,BBUF
1505 D3FB C55BC660C4         FDB SSMOD,FROMR,PLUS
```

```
1506 D401 0638                FDB BLOCK-RAM+BRAM
1507 D403 C459D142C0          FDB PLUS,CSL,SEMIS
1508 D409 852E4C494E          FCB $85,'.,'L,'I,'N
1509 D40E C5                  FCB 'E+$80
1510 D40F D3EA                FDB PLINE-9
1511 D411 C04ED3F3    DOTLIN  FDB DOCOL,PLINE
1512 D415 CBB1CB83C0          FDB DTRAIL,TYPE,SEMIS
1513 D41B 844C4953D4          FCB $84,'L,'I,'S,'T+$80
1514 D420 D409                FDB DOTLIN-8
1515 D422 C04ECB2AC1  LIST    FDB DOCOL,DEC,CR,DUP,SCR
1516 D42C C6D8CBE4            FDB STORE,PDOTQ
1517 D430 0653435220          FCB 6,'S,'C,'R,32,'#,32
1518 D437 D32DD12CD1          FDB DOT,BBUF,CSL,SLASH
1519 D43F C784C23C            FDB ZERO,XDO
1520 D443 C112C24BC7  LIST2   FDB CR,I,TWO,DOTR,SPACE
1521 D44D C24BC86EC6          FDB I,SCR,AT,DOTLIN
1522 D455 C0FDC1E6            FDB QTERM,ZBRAN
1523 D459 0004C644            FDB LIST3-*,LEAVE
1524 D45D C203        LIST3   FDB XLOOP
1525 D45F FFE4C112C0          FDB LIST2-*,CR,SEMIS
1526 D465 85434C4541          FCB $85,'C,'L,'E,'A
1527 D46A D2                  FCB 'R+$80
1528 D46B D41B                FDB LIST-7
1529 D46D C04E0638    CLEAR   FDB DOCOL,BLOCK-RAM+BRAM
1530 D471 D12CCD63C0          FDB BBUF,BLANKS,SEMIS
1531 D477 84544F4EC5          FCB $84,'T,'O,'N,'E+$80
1532 D47C D465                FDB CLEAR-8
1533 D47E D480        TONE    FDB *+2
1534 D480 3434                PSHS B,X,Y
1535 D482 3720                PULU Y
1536 D484 3706                PULU D
1537 D486 D78C                STB $8C
1538 D488 1F20                TFR Y,D
1539 D48A 3440                PSHS U
1540 D48C BDA951              JSR $A951
1541 D48F 3540                PULS U
1542 D491 3534                PULS B,X,Y
1543 D493 16EBBC              LBRA NEXT
1544 D496 B6FF01      NCLR    LDA $FF01
1545 D499 8A08                ORA #$08
1546 D49B B7FF01              STA $FF01
1547 D49E 863C                LDA #$3C
1548 D4A0 B7FF23              STA $FF23
1549 D4A3 8620                LDA #$20
1550 D4A5 8E01D1              LDX #$1D1
1551 D4A8 A780        NCLR2   STA ,X+
1552 D4AA 8C01DA              CMPX #$1DA
1553 D4AD 26F9                BNE NCLR2
1554 D4AF 8601                LDA #$01
1555 D4B1 B701D1              STA $1D1
1556 D4B4 39                  RTS
1557 D4B5 8557524954          FCB $85,'W,'R,'I,'T
```

```
1558 D4BA C5                    FCB 'E+$80
1559 D4BB D477                  FDB TONE-7
1560 D4BD C04E0638      WRITE   FDB DOCOL,BLOCK-RAM+BRAM
1561 D4C1 D4C5C05B              FDB BWRIT,SEMIS
1562 D4C5 D4C7          BWRIT   FDB *+2
1563 D4C7 3434                  PSHS B,X,Y
1564 D4C9 BDD496                JSR NCLR
1565 D4CC 0F78                  CLR $78
1566 D4CE 3440                  PSHS U
1567 D4D0 8601                  LDA #$01
1568 D4D2 BDA65C                JSR $A65C
1569 D4D5 3540                  PULS U
1570 D4D7 3710                  PULU X
1571 D4D9 31890400              LEAY $400,X
1572 D4DD 109F02                STY $02
1573 D4E0 A680          WRIT3   LDA  ,X+
1574 D4E2 BDA290                JSR $A290
1575 D4E5 9C02                  CMPX $02
1576 D4E7 26F7                  BNE WRIT3
1577 D4E9 9679                  LDA $79
1578 D4EB 2703                  BEQ WRIT4
1579 D4ED BDA2A8                JSR $A2A8
1580 D4F0 BDA444        WRIT4   JSR $A444
1581 D4F3 3534                  PULS B,X,Y
1582 D4F5 16EB5A                LBRA NEXT
1583 D4F8 845245 41C4           FCB $84,'R,'E,'A,'D+$80
1584 D4FD D4B5                  FDB WRITE-8
1585 D4FF C04E0638      READ    FDB DOCOL,BLOCK-RAM+BRAM
1586 D503 D507C05B              FDB BREAD,SEMIS
1587 D507 D509          BREAD   FDB *+2
1588 D509 3434                  PSHS B,X,Y
1589 D50B BDD496                JSR NCLR
1590 D50E BEC179                LDX WENT
1591 D511 BF018F                STX $18F
1592 D514 867E                  LDA #$7E
1593 D516 B7018E                STA $18E
1594 D519 3710                  PULU X
1595 D51B 9F02                  STX $02
1596 D51D 0F78                  CLR $78
1597 D51F 3440                  PSHS U
1598 D521 BDA629                JSR $A629
1599 D524 0D79          READ2   TST $79
1600 D526 270B                  BEQ READ3
1601 D528 BDA186                JSR $A186
1602 D52B 9E02                  LDX $02
1603 D52D A780                  STA ,X+
1604 D52F 9F02                  STX $02
1605 D531 20F1                  BRA READ2
1606 D533 0F78          READ3   CLR $78
1607 D535 3540                  PULS U
1608 D537 3534                  PULS B,X,Y
1609 D539 16EB16                LBRA NEXT
```

```
1610 D53C 8653435245          FCB $86,'S,'C,'R,'E,'E
1611 D542 CE                  FCB 'N+$80
1612 D543 D4F8                FDB READ-7
1613 D545 C7630400    SCREEN  FDB DOCON,$0400
1614 D549 83434CD3            FCB $83,'C,'L,'S+$80
1615 D54D D53C                FDB SCREEN-9
1616 D54F D551        CLS     FDB *+2
1617 D551 3434                PSHS B,X,Y
1618 D553 3706                PULU D
1619 D555 BDA91C              JSR $A91C
1620 D558 3534                PULS B,X,Y
1621 D55A 16EAF5              LBRA NEXT
1622 D55D 844A5354CB          FCB $84,'J,'S,'T,'K+$80
1623 D562 D549                FDB CLS-6
1624 D564 D566        JSTK    FDB *+2
1625 D566 3474                PSHS B,X,Y,U
1626 D568 BDA9DE              JSR $A9DE
1627 D56B 3574                PULS B,X,Y,U
1628 D56D 16EAE2              LBRA NEXT
1629 D570             RAM     EQU *
1630 D570 872841424F          FCB $87,'(,'A,'B,'O,'R,'T
1631 D577 A9                  FCB ')+$80
1632 D578 D55D                FDB JSTK-7
1633 D57A C04ED0DCC0  PABORT  FDB DOCOL,ABORT,SEMIS
1634 D580 85424C4F43          FCB $85,'B,'L,'O,'C
1635 D585 CB                  FCB 'K+$80
1636 D586 0620                FDB PABORT-10-RAM+BRAM
1637 D588 C04ED389C0  BLOCK   FDB DOCOL,PBLOCK,SEMIS
1638 D58E C5464F5254          FCB $C5,'F,'O,'R,'T
1639 D593 C8                  FCB 'H+$80
1640 D594 0630                FDB BLOCK-8-RAM+BRAM
1641 D596 C719D07581  FORTH   FDB DODOES,DOVOC,$81A0
1642 D59C 06660000            FDB EDITOR-9-RAM+BRAM,0
1643 D5A0 C9                  FCB $C9
1644 D5A1 415353454D          FCC /ASSEMBLE/
1645 D5A9 D2                  FCB 'R+$80
1646 D5AA 063E                FDB FORTH-8-RAM+BRAM
1647 D5AC C719D07581  ASSEM   FDB DODOES,DOVOC,$81A0
1648 D5B2 064A0000            FDB FORTH+4-RAM+BRAM,0
1649 D5B6 C645444954          FCB $C6,'E,'D,'I,'T,'O
1650 D5BC D2                  FCB 'R+$80
1651 D5BD 0650                FDB ASSEM-12-RAM+BRAM
1652 D5BF C719D07581  EDITOR  FDB DODOES,DOVOC,$81A0
1653 D5C5 064A0000            FDB FORTH+4-RAM+BRAM,0
1654 D5C9 D5C9                FDB *
1655 D5CB 0000        ERAM    FDB 0
```

1656 D5CD END KRNL

ABORT	D0DC	ABS	C59A	ABS2	C5A6	ALLOT	C8FA
AND	C429	ASSEM	D5AC	AT	C6C0	BACK	D1B0
BASE	C8AE	BBUF	D12C	BCOMP	CF97	BDIGS	D26A
BEGIN	D1C2	BL	C7A5	BLANKS	CD63	BLK	C851
BLOCK	D588	BMAX	C809	BRAM	0620	BRAN	C1DA
BREAD	D507	BSCR	D138	BUILDS	C6FD	BWRIT	D4C5
CAT	C6CC	CCOMM	C917	CEMIT	C0B7	CENT	C12A
CFA	C9EA	CKEY	C0E2	CLEAR	D46D	CLS	D54F
CMOV2	C328	CMOV3	C332	CMOVE	C31A	COLD	C128
COLD2	C135	COLDZ	C154	COLON	C03A	COMMA	C906
COMPIL	CAC9	CON	C759	CONTXT	C889	COUNT	CB70
CR	C112	CREAT2	CF55	CREATE	CF3B	CSL	D142
CSP	C8CC	CSTORE	C6E8	CURENT	C897	DABS	C5AF
DABS2	C5BB	DDOT	D31F	DDOTR	D2EB	DDUP	C98E
DDUP2	C998	DEC	CB2A	DEFIN	D08D	DFIND	CE91
DFIND2	CEB1	DIG	D2A8	DIG2	D2C4	DIGIT	C25A
DIGIT0	C270	DIGIT1	C278	DIGIT2	C27D	DIGS	D2D3
DIGS2	D2D5	DLITE2	CFE0	DLITER	CFD0	DMINUS	C49D
DMINX	C4B3	DO	D1ED	DOCOL	C04E	DOCON	C763
DODOES	C719	DOES	C70D	DOT	D32D	DOTLIN	D411
DOTQ	CBFD	DOTQ1	CC1D	DOTQ2	CC25	DOTR	D30E
DOUSER	C7DC	DOVAR	C779	DOVOC	D075	DP	C838
DPINIT	C00A	DPL	C8B8	DPLUS	C467	DROP	C684
DSETS2	C610	DSETSN	C606	DTRAIL	CBB1	DTRAL2	CBB9
DTRAL3	CBD3	DTRAL4	CBD7	DUP	C6A3	EDIGS	D279
EDITOR	D5BF	ELSE	D22A	EMIT	C09D	EMIT2	C0B3
ENCL11	C303	ENCL2	C307	ENCL4	C30B	ENCL01	C2E9
ENCL02	C2F9	ENCL05	C2DD	ENDIF	D1D4	EQUAL	C933
ERAM	D5CB	ERASE	CD52	ERROR	CEBB	ERROR2	CEC9
EXEC	C06C	EXPEC2	CC7F	EXPEC3	CCAB	EXPEC4	CCC3
EXPEC5	CCC5	EXPEC6	CCD1	EXPECT	CC75	FENCE	C82F
FENCIN	C008	FILL	CD32	FIRST	C7B1	FLD	C8C2
FORGET	D173	FORTH	D596	FOUND	C2C4	FROMR	C660
GETX	C030	GREAT	C93F	HERE	C8EA	HEX	CB14
HLD	C8DF	HOLD	CD72	I	C24B	IDDOT	CEEC
IF	D213	IMMED	D03C	IN	C85A	INTER2	CFF0
INTER3	D008	INTER4	D00C	INTER5	D012	INTER6	D026
INTER7	D02A	INTERP	CFEE	JSTK	D564	KEY	C0D6
KRNL	C000	LATEST	C9CA	LBRAK	CADF	LCASE	C35F
LEAVE	C644	LESS	C5C1	LESSF	C5CF	LESST	C5D2
LESSX	C5D4	LFA	C9DA	LIMIT	C7BD	LIST	D422
LIST2	D443	LIST3	D45D	LIT	C1CA	LITER	CFB3
LITER2	CFC3	LOAD	D3C2	LOOP	D200	MAX	C975
MAX2	C983	MAXBLK	C012	MESS	D150	MEXIT	C090
MIN	C95D	MIN2	C96B	MINUS	C484	MINUS2	C48F
MOD	C54B	MSLASH	C502	MSMOD	C57E	MSTAR	C4D9
MTEXT	C07A	MTEXT2	C082	N	06A0	NBLK	0008
NCLR	D496	NCLR2	D4A8	NEXT	C052	NEXT3	C054
NFA	C9F8	NMTCH	C08F	NULL	CCFB	NULL1	CD21

NULL2	CD25	NULL3	CD29	NUMB	CE39	NUMB1	CE57
NUMB2	CE7D	NUMB3	CE87	OFSET	C87B	ONE	C78C
ONEP	C4BB	OR	C43B	OUT	C864	OVER	C676
PA	06A4	PA0	06A2	PABORT	D57A	PAD	CD8A
PAREN	D09D	PBLOCK	D389	PCHR	06A6	PD	06A0
PDOTQ	CBE4	PFA	CA0E	PFIND	C28D	PFIND0	C291
PFIND1	C297	PFIND2	C2A6	PFIND3	C2B2	PFIND4	C2BE
PFIND8	C2BA	PFINDE	C2CE	PLINE	D3F3	PLUS	C459
PNUMB	CDEE	PNUMB2	CDF0	PNUMB3	CE26	PNUMB4	CE2C
PORIG	C7EE	PRGBGN	C000	PSCODE	CB40	PSEMIS	C05D
PSTORE	C6AF	PULLDX	C02A	PUSHD	C032	PUTD	C431
QCOMP	CA50	QCSP	CA92	QERR	CA36	QERR2	CA44
QERR3	CA46	QEXEC	CA68	QLOAD	CAAF	QPAIRS	CA7F
QSTACK	CC30	QTERM	C0FD	QUERY	CCE3	QUIT	D0AE
QUIT2	D0B8	QUIT3	D0D0	R	C66D	RAM	D570
RBRAK	CAED	READ	D4FF	READ2	D524	READ3	D533
RINIT	C01E	RNUM	C8D5	ROT	CC60	RPSTOR	C1BB
SCR	C86E	SCREEN	D545	SCSP	CA23	SEMI	C740
SEMIC	CB56	SEMIS	C05B	SETSN	C5F4	SETSN2	C5FE
SIGN	D292	SIGN2	D2A2	SINIT	C01A	SLASH	C53B
SLMOD	C52B	SMOV2	C34F	SMOV3	C363	SMOV4	C35B
SMOVE	C341	SMUDGE	CB02	SPACE	C94F	SPACE2	D25D
SPACE3	D263	SPACES	D24D	SPAT	C19E	SPSTOR	C1AD
SSLASH	C56C	SSMOD	C55B	STAR	C4F5	STATE	C8A3
STOD	C5DF	STOD2	C5EA	STORE	C6D8	STOREX	C02C
SUB	C927	SWAP	C692	THREE	C79C	TIB	C7FE
TIBINT	C01C	TICK	D15A	TOGGLE	C72E	TONE	D47E
TOPDEF	C00E	TOPEDT	C010	TOR	C652	TRAV	C9A5
TRAV2	C9A9	TSLS	C41A	TSTR	C40C	TWO	C794
TWOP	C4CA	TYPE	CB83	TYPE2	CB93	TYPE3	CBA1
TYPE4	CBA3	UORIG	06AC	UP	06AA	UPINIT	C006
USER	C7D6	USLASH	C3D9	USLL1	C3EA	USLL2	C3FA
USRBGN	06A0	USREND	09A0	USTAR	C3A3	VAR	C773
VIRBGN	09A0	VIREND	29C0	VLIST	D33D	VLIST1	D345
VLIST2	D363	VOCAB	D055	VOCINT	C00C	VOCLIN	C847
WARM	C177	WARM2	C17F	WARN	C823	WAVE	C371
WAVE2	C386	WAVE3	C397	WAVE4	C391	WENT	C179
WIDINT	C024	WIDTH	C815	WORD	CD9D	WORD2	CDB1
WORD3	CDB5	WRIT3	D4E0	WRIT4	D4F0	WRITE	D4BD
WRNINT	C028	XDO	C23C	XLOOP	C203	XOR	C44B
XPLOF	C229	XPLONO	C231	XPLOOP	C214	XPLOP2	C218
XVIRBG	C016	XVIRED	C018	ZBNO	C1F5	ZBRAN	C1E6
ZBYES	C1EC	ZEQU	C617	ZEQU2	C620	ZERO	C784
ZLESS	C62A	ZLESS2	C638				

```
SCR 1
 0 ( COLORFORTH 1.0 )
 1 ( READ  WRITE WORDS ) DECIMAL
 2 : READS OVER + SWAP
 3         DO I . I READ LOOP ;
 4 : CLOADS 0
 5       DO 1 READ 1 LOAD
 6       LOOP ;
 7 : WRITES OVER + SWAP
 8       DO I . I WRITE LOOP ;
 9 : +LOOP 3 ?PAIRS COMPILE (+LOOP)
10           BACK ; IMMEDIATE
11 HEX
12  88 CONSTANT CUR
13  8D CONSTANT TIME
14 : TIME0 -1 TIME ! ;
15 : ?TIME -1 TIME @ - ;
16 : THRU 1+ SWAP
17        DO I . I LOAD LOOP ;
18 : MOTOR FF21 8 TOGGLE ;
19 : ? @ . ;
20 : B <BUILDS , DOES> @ 95 ! ;
21 01F3 B BR110
22 00B4 B BR300
23 0057 B BR600
24 0029 B BR1200
25 0012 B BR2400
26 81A0 ' B NFA !
27 6F CONSTANT CHAN
28 DECIMAL ;S
29
30
31
```

```
SCR 2
 0 ( COLORFORTH 1.0 ) DECIMAL
 1 : UNTIL 1 ?PAIRS COMPILE 0BRANCH
 2         BACK ; IMMEDIATE
 3 : AGAIN 1 ?PAIRS COMPILE BRANCH
 4         BACK ; IMMEDIATE
 5 : REPEAT >R >R [COMPILE] AGAIN
 6          R> R> 2 - [COMPILE]
 7          ENDIF ; IMMEDIATE
 8 : WHILE [COMPILE] IF 2+ ;
 9         IMMEDIATE
10 : TEXT HERE C/L 1+ BLANKS WORD
11        HERE PAD C/L 1+ CMOVE ;
12 HEX
13 : LINE DUP FFE0 AND 17 ?ERROR
14         SCR @ (LINE) DROP ;
15 : 2DUP OVER OVER ;
16 : 2DROP DROP DROP ;
17 : 2SWAP ROT >R ROT R> ;
18 : 2OVER >R >R 2DUP
19      R> R> 2SWAP ;
20 : TRY SCR @ LOAD ;
21 : B  DO I OVER 1 AND
22           IF 1+ ENDIF
23           0 SWAP C! 2/ 2
24      +LOOP DROP ;
25 : PG! FFD2 FFC6 B ;
26 : VDG!  FFC6 FFC0 B ;
27 81A0 ' B NFA !
28 : VCR!  8 * FF22 C@ 7 AND OR
29         FF22 C! ;
30 : VNORM 0 VDG! 0 VCR! 2 PG! ;
31 DECIMAL ;S
```

```
SCR 3
 0 ( COLORFORTH 1.0 )
 1 HEX
 2 : B <BUILDS , DOES> @ C@ ;
 3 15A B J0
 4 15B B J1
 5 15C B J2
 6 15D B J3
 7 81A0 ' B NFA !
 8 ( GET AMOUNT OF MEMORY )
 9 : FREE 74 @ HERE - ;
10 : U. 0 D. ;
11 : H. BASE @ SWAP HEX U.
12       BASE ! ;
13 : FORM 0C EMIT ;
14 : PRINT -2 CHAN C! INTERPRET CR
15           0 CHAN C! ;
16 : DUMP OVER + SWAP CR
17       DO I 0 5 D.R SPACE I 8 + I
18            DO I C@ 3 .R
19            LOOP CR 8
20      +LOOP ;
21 : THEN [COMPILE] ENDIF ;
22       IMMEDIATE
23 : ASC 0A /MOD 30 + 100 * SWAP
24       30 + + ;
25 : TICKS
26       BEGIN ?TIME OVER < 0=
27       UNTIL DROP ;
28 DECIMAL ;S
29
30
31
```

```
SCR 4
 0 ( COLORFORTH 1.0 )
 1 ( VIEW ) HEX
 2 EDITOR DEFINITIONS
 3 : #LOCATE R# @ C/L /MOD ;
 4 : V  SCR @ BLOCK R# @
 5   C/L / 5 - 0 MAX 15 MIN
 6   C/L * + SCREEN
 7   160 SMOVE
 8   #LOCATE DUP
 9   15 - 5 MAX MIN
10   C/L * + SCREEN + 40 TOGGLE
11   560 C/L BLANKS
12   SCR @   ASC 560 !
13   #LOCATE ASC 563 !
14           ASC 566 ! ;
15 : Z DUP ;
16 : ?31 0 1F LINE C/L OVER + SWAP
17       DO I C@ BL -
18         IF DROP 1 LEAVE ENDIF
19       LOOP
20   IF ." DATA IN 31, QUIT Y/N"
21      KEY 59 = IF Z ENDIF
22   ENDIF ;
23 : BMOV PAD 1+ C@ IF  PAD
24      SWAP C/L 1+ CMOVE ELSE
25      DROP ENDIF ;
26 : PADF PAD 50 + ;
27 : PADI PADF 50 + ;
28 : !CUR 0 MAX B/BUF 1 - MIN
29        R# ! ;
30 DECIMAL ;S
31
```

```
SCR 5
 0 ( COLORFORTH 1.0 )
 1 EDITOR DEFINITIONS
 2 : >L#    #LOCATE SWAP DROP ;
 3 : #LEAD #LOCATE LINE SWAP ;
 4 : #LAG  #LEAD DUP >R +
 5         C/L R> - ;
 6 : -MOVE LINE C/L CMOVE  ;
 7 : H >L# LINE PADI 1+ C/L DUP
 8     PADI C! CMOVE ;
 9 : K >L# LINE C/L BL FILL ;
10 : SP ?31 >L# 31 DO I 1 -
11     LINE I -MOVE -1 +LOOP K ;
12 : X >L# DUP H 31 <
13     IF 31 SWAP
14         DO I 1+ LINE I -MOVE
15         LOOP
16     ELSE DROP ENDIF
17     31 LINE C/L 32 FILL ;
18 : C R# @ + !CUR ;
19 : T C/L * !CUR ;
20 : +T >L# + T ;
21 : J 1 +T ;
22 ( CHANGE J TO DOWN ARROW )
23 HEX  8A ' J  NFA 1+ C!
24 : JJ 4 +T ;
25 ( CHANGE JJ TO DOUBLE DOWN )
26     A8A ' JJ NFA 1+ !
27 : ↑  -1 +T ;
28 : ↑↑ -4 +T ;
29 DECIMAL ;S
30
31
```

```
SCR 6
 0 ( COLORFORTH 1.0 )
 1 HEX
 2 : GTEXT 5E TEXT BMOV ;
 3 : (R) >L# PADI 1+ SWAP -MOVE ;
 4 : P PADI GTEXT (R) ;
 5 : U 1 +T SP P ;
 6 : I 5 C ;
 7 ( CHANGE I TO RIGHT ARROW )
 8 8189 ' I NFA !
 9 : TOP 0 R# ! ;
10 : BOT B/BUF C/L - !CUR ;
11 : COPY BLOCK SWAP BLOCK SWAP
12   B/BUF CMOVE  ;
13 : MATCH >R >R 2DUP R> R>
14   2SWAP OVER + SWAP
15   DO 2DUP FORTH I -TEXT
16     IF >R 2DROP R> - I SWAP -
17       0 SWAP 0 0 LEAVE
18     ENDIF
19   LOOP 2DROP SWAP 0= SWAP ;
20 : 1LINE #LAG PADF COUNT MATCH
21         R# +! ;
22
23 : SERR TOP PADF HERE C/L 1+
24     CMOVE HERE COUNT TYPE
25     7 EMIT ."  NONE" Z ;
26 : (F)
27     BEGIN 3FF R# @ <
28         IF SERR ENDIF 1LINE
29     UNTIL ;
30
31 DECIMAL ;S
```

```
SCR 7
 0 ( COLORFORTH 1.0 ) HEX
 1 : DEL R# @ OVER < 17 ?ERROR
 2       #LAG SWAP DROP C/L =
 3       IF -1 C [ SMUDGE ] 1 DEL
 4          1 C 1 -
 5       ENDIF >R #LAG + FORTH R -
 6        #LAG R MINUS R# +!
 7        #LEAD + SWAP CMOVE
 8        R> -DUP [ SMUDGE ]
 9     IF BLANKS ELSE DROP ENDIF ;
10 : F PADF GTEXT (F) ;
11 : E PADF C@ DEL ;
12 : D F E ;
13 : TILL PADF GTEXT
14         #LEAD + 1LINE 0=
15            IF SERR ENDIF
16         #LEAD + SWAP - DEL ;
17 : (I) PADI COUNT #LAG ROT OVER
18       MIN C 2DUP PADI COUNT +
19       SWAP CMOVE CMOVE  ;
20 : I PADI GTEXT (I) ;
21 : R E I ;
22 : N 1 SCR +! TOP ;
23 : B -1 SCR +! TOP ;
24 : M OVER BMAX @ > 8 ?ERROR
25     SCR @ >R R# @ >R H SWAP
26       SCR ! 1+ C/L * !CUR
27      SP (R) R> C/L + !CUR
28       R> SCR ! ;
29 : L SCR @ LIST ;
30 : WIPE SCR @ CLEAR  ;
31 DECIMAL ;S
```

```
SCR 8
 0 ( COLORFORTH 1.0 ) HEX
 1 : RANGE  1 MAX BMAX @ MIN ;
 2 : SCROL 60 CUR @ C!
 3         5A0 580 60 CMOVE
 4         5E0 C/L 60 FILL
 5         5E0 CUR ! ;
 6 : NEW SCROL
 7       BEGIN QUERY PADI
 8             GTEXT PAD 1+ C@
 9       WHILE (R) 1 +T U SCROL
10       REPEAT ;
11 : QIT BEGIN RP! [COMPILE] EDITOR
12       SCR @ RANGE SCR ! 0 BLK !
13           ."  [OK]" SCROL
14           U QUERY INTERPRET
15      AGAIN ;
16 ' QIT CFA ' Z ! ( LINK Z>QIT )
17 81A0 ' Z NFA ! ( REMOVE LINK )
18 : ERR HERE COUNT TYPE
19      ." <-?" MESSAGE SP!
20      7 EMIT [COMPILE] [ QIT ;
21 : QUIT 1 WARNING !
22         ' ABORT CFA ' (ABORT) !
23         CURRENT @ CONTEXT !
24         ." DONE" QUIT ;
25   FORTH DEFINITIONS
26 : EDIT [COMPILE] EDITOR EDITOR
27         ' ERR CFA ' (ABORT) !
28         -1 WARNING ! RANGE SCR !
29         TOP QIT ;
30 FORTH HEX HERE H. DECIMAL ;S
31
```

```
SCR 10
 0 ( COLORFORTH 1.0 )
 1 HEX
 2 EDITOR DEFINITIONS
 3 : S  PADF GTEXT
 4      SCR @ DUP >R FORTH
 5      DO I SCR ! TOP
 6           BEGIN 1LINE
 7                IF U SCROL
 8                     ." FOUND"
 9                     KEY 3 =
10                     IF QIT
11                     ENDIF
12                ENDIF
13                  3FF R# @ <
14           UNTIL
15      LOOP R> SCR ! TOP ;
16 FORTH DEFINITIONS
17 DECIMAL ;S
18
19
20
21
22
23
24
25
26
27
28
29
30
31
```

```
SCR 11
 0 ( COLORFORTH 1.0 )
 1 DECIMAL
 2 : ?DUP -DUP ;
 3 : NEGATE MINUS ;
 4 : >IN IN ;
 5 : EXIT R> DROP ;
 6 : NOT 0= ;
 7 : CONVERT (NUMBER) ;
 8 TIB CONSTANT S0
 9 : DNEGATE DMINUS ;
10 : DEPTH SP@ S0 @ SWAP - 2/ ;
11 : D- DMINUS D+ ;
12 : 1- 1 - ;
13 : D0= OR 0= ;
14 : LISTS OVER + SWAP
15      DO I LIST FORM
16      LOOP ;
17 : ASCII BL WORD HERE 1+ C@
18      [COMPILE] LITERAL ;
19      IMMEDIATE
20
21 DECIMAL ;S
22
23
24
25
26
27
28
29
30
31
```

```
SCR 12
 0 ( COLORFORTH 1.0 )
 1 HEX
 2 48 USER COLOR
 3 4A USER GSCR
 4    0 CONSTANT GREEN
 5 4000 CONSTANT YELLOW
 6 8000 CONSTANT BLUE
 7 C000 CONSTANT RED
 8 : D< ROT OVER OVER =
 9      IF ROT ROT D- 0< SWAP DROP
10      ELSE SWAP < SWAP DROP SWAP
11           DROP
12      ENDIF ;
13 : U< 0 SWAP 0 D< ;
14 : 2@ DUP 2+ @ SWAP @ ;
15 : 2! OVER OVER ! 2+
16      SWAP DROP ! ;
17 : 2CONSTANT <BUILDS , ,
18      DOES> 2@ ;
19 : 2VARIABLE VARIABLE , ;
20 : D= D- D0= ;
21 : DMAX 2OVER 2OVER D<
22      IF 2SWAP ENDIF 2DROP ;
23 : DMIN 2OVER 2OVER D<
24      IF 2DROP
25      ELSE 2SWAP 2DROP
26      ENDIF ;
27 : 0> 0 > ;
28 : R@ R> R SWAP >R ;
29 : 'S SP@ ;
30 : PAGE 1 CLS ;
31 DECIMAL ;S
```

```
SCR 13
0 ( COLORFORTH 1.0 )
1 ( SOURCE AND DECODE ) DECIMAL
2 ( CONDITIONALL PRINT A CR )
3 : ?CR OUT @ 25 > IF CR ENDIF ;
4 ( SEARCH FOR NAME IN DICT )
5 ( PRINT BYTE IF NOT FOUND )
6 ( ADDR1 --- )
7 : .NAME LATEST
8       BEGIN DUP >R PFA CFA OVER =
9             IF R> ?CR ID.
10            ELSE R> PFA LFA @ -DUP
11                 IF 0
12                 ELSE 0 256 U/
13                      . DROP 1 - 1
14                 ENDIF
15            ENDIF
16      UNTIL ;
17
18 ( RETURN ADDRESS OF ;S )
19 ( --- ADDR1 )
20 ' ;S CFA CONSTANT ';S
21 DECIMAL ;S
22
23 HOLD DOWN ANY KEY TO STOP !!
24
25
26
27
28
29
30
31
```

```
SCR 14
 0 ( COLORFORTH 1.0 )
 1 HEX
 2
 3 4C USER AFLG
 4
 5 DECIMAL
 6
 7 : AON 1 AFLG ! ;
 8
 9 : AOFF 0 AFLG ! ;
10
11 ( ADDR1 --- )
12 : ASOURCE 2 -
13       BEGIN 2+ AFLG @
14           IF CR DUP U. ENDIF
15             DUP @ .NAME DUP @ ';'S
16             = ?TERMINAL OR
17       UNTIL DROP ;
18
19 ( --- ) ( FOLLOW BY STRING )
20 : SOURCE CR ." : "
21       [COMPILE] ' DUP NFA ID.
22       ASOURCE ;
23
24 DECIMAL ;S
25
26
27
28
29
30
31
```

```
SCR 15
 0 ( COLORFORTH 1.0 )
 1 ( NEW S. ) DECIMAL
 2
 3 FORTH DEFINITIONS
 4
 5 : .S DEPTH -DUP
 6       IF 1+ 1
 7          DO S0 @ I 2* - @ .
 8          LOOP
 9       ELSE 1 MESSAGE
10       ENDIF ;
11 DECIMAL ;S
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
```

```
SCR 16
 0 ( COLORFORTH 1.0 )
 1 ( NEW MESSAGE ) DECIMAL
 2 : MESS DUP
 3   0 = IF ." WHAT"          ENDIF DUP
 4   1 = IF ." STACK EMPTY" ENDIF
 5                 DUP
 6   2 = IF ." MEM FULL"    ENDIF DUP
 7   8 = IF ." BLK RANGE"   ENDIF DUP
 8  17 = IF ." ?COMPILE"    ENDIF DUP
 9  18 = IF ." ?EXECUTE"    ENDIF DUP
10  19 = IF ." ?PAIRS"      ENDIF DUP
11  20 = IF ." NOT DONE"    ENDIF DUP
12  21 = IF ." SAVED VOC"   ENDIF DUP
13  22 = IF ." ?LOADING"    ENDIF DUP
14  23 = IF ." OFFSCREEN"   ENDIF DUP
15  24 = IF ." SET VOCAB"   ENDIF
16  25 = IF ." BAD OVERLAY" ENDIF ;
17
18   ' MESS CFA ' MESSAGE !
19 DECIMAL ;S
20
21
22
23
24
25
26
27
28
29
30
31
```

```
SCR 17
 0 ( COLORFORTH 1.0 )
 1 ( CODE DEFINITION ) HEX
 2 : CODE ?EXEC CREATE
 3       [COMPILE] ASSEMBLER !CSP ;
 4       IMMEDIATE
 5 ' FORTH 2+ ' ASSEMBLER 4 + !
 6
 7 ASSEMBLER DEFINITIONS
 8 : C; CURRENT @ CONTEXT !
 9       ?EXEC ?CSP SMUDGE ;
10       IMMEDIATE
11 : NEXT AEA1 , 6E94 , ;
12
13 FORTH DEFINITIONS
14 CODE I'
15       ( LDD 4,S )   EC64 ,
16       ( PUSU D  )   3606 ,
17         NEXT
18  C;
19 CODE J
20       ( LDD 8,S )   EC68 ,
21       ( PUSU D  )   3606 ,
22       ( NEXT    )   AEA1 , 6E94 ,
23  C;
24
25
26
27 DECIMAL ;S
28
29
30
31
```

```
SCR 18
 0 ( COLORFORTH 1.0 )
 1 ( OVERLAYS ) HEX
 2
 3 : ?OV LIMIT @ 81A0 -
 4       19 ( 25 DEC ) ?ERROR ;
 5
 6 : EMPTY [COMPILE] FORTH
 7       DEFINITIONS
 8       C00E @ ' FORTH 4 + !
 9       C010 @ ' EDITOR 4 + !
10       ASSEMBLER ' NEXT NFA
11       ' ASSEMBLER 4 + !
12       LIMIT DP ! ;
13
14 : #BLOCKS EMPTY 1 MAX
15       74 @ 400 / 2 - MIN BMAX !
16       LIMIT DP ! ;
17
18 : OVLINK ?OV LIMIT PFA DUP LFA
19       LATEST SWAP ! @
20       CURRENT @ ! LIMIT PFA 2+ @
21       DP ! ;
22
23 : OVSTART EMPTY HERE 81A0 ,
24       LATEST , CURRENT @ !
25       BMAX @ , 0 , 0 , 0 , ;
26
27 DECIMAL ;S
28
29
30
31
```

```
SCR 19
 0 ( COLORFORTH 1.0 )
 1 ( OVERLAYS ) DECIMAL
 2
 3 : OVEND ?OV LIMIT PFA LATEST
 4       OVER !
 5       2+ HERE OVER ! 2+ HERE
 6       LIMIT - B/BUF /MOD SWAP
 7       IF 1+
 8       ENDIF SWAP ! ;
 9
10 : OVSAVE ?OV LIMIT PFA 4 + @ 0
11       DO 1 BLOCK I B/BUF * LIMIT
12          + SWAP B/BUF CMOVE
13          1 WRITE
14       LOOP ;
15
16 : OVLOAD 1 READ 1 BLOCK
17       DUP PFA CFA @ #BLOCKS
18       DUP LIMIT B/BUF CMOVE ?OV
19       PFA 4 + @ 1 - -DUP
20       IF 1+ 1
21          DO 1 READ 1 BLOCK I
22             B/BUF * LIMIT +
23             B/BUF CMOVE
24          LOOP
25       ENDIF OVLINK ;
26
27 DECIMAL ;S
28
29
30
31
```

```
SCR 20
 0 ( COLORFORTH 1.0 )
 1 HEX
 2 : GMODE <BUILDS C, C, DOES>
 3       COUNT VDG! C@ VCR! ;
 4 14 2 GMODE G2C
 5 18 4 GMODE G3C
 6 1C 6 GMODE G6C
 7 CODE SET
 8       3706 , 8620 , 3DDD ,
 9        02 C, 3706 ,  B6 C,
10      COLOR , 5424 , 0244 ,
11       4454 , 2404 , 4444 ,
12       4444 ,  97 C,  04 C,
13       4FD3 ,  02 C,  F3 C,
14       GSCR , 1F01 , A684 ,
15        98 C,  04 C, A784 ,
16       NEXT C;
17 DECIMAL ;S
18
19
20
21
22
23
24
25
26
27
28
29
30
31
```

```
SCR 21
 0 ( COLORFORTH 1.0 )
 1 HEX
 2 FORTH DEFINITIONS
 3
 4 : TASK ;
 5 ' TASK NFA C00E !
 6
 7 EDITOR
 8 ' S NFA C010 !
 9
10 ASSEMBLER
11 ' NEXT NFA D5B2 !
12
13 FORTH DEFINITIONS
14 DECIMAL ;S
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
```

```
SCR 22
 0 ( COLORFORTH 1.0 )
 1 DECIMAL
 2 : DLIST
 3       CR CR CR CR CR
 4       DECIMAL CR DUP SCR !
 5    ."     SCR " DUP . 32 SPACES
 6       ." SCR " 1+ . B/BUF C/L / 0
 7       DO CR 3 SPACES
 8          I 2 .R SPACE I SCR @
 9          (LINE) TYPE 3 SPACES
10          I 2 .R SPACE
11          I SCR @ 1+ .LINE
12       LOOP CR FORM ;
13 : LTHRU 1+ SWAP
14       DO I DLIST 2
15      +LOOP ;
16 DECIMAL ;S
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
```

```
SCR 23
 0 DECIMAL
 1 EDITOR DEFINITIONS
 2 : GET PAD C/L 1+ BLANKS EDITOR
 3         #LEAD 0 SWAP FORTH
 4         DO I OVER + C@ BL =
 5            IF I + LEAVE ENDIF -1
 6        +LOOP EDITOR
 7         #LAG 0 FORTH
 8         DO I OVER + C@ BL =
 9            IF I + LEAVE ENDIF
10         LOOP OVER - PAD 2DUP C!
11         1+ SWAP CMOVE
12         PAD C@ 0=
13         IF BL 256 + PAD !
14         ENDIF PADI BMOV ;
15
16 : W  GET PADF BMOV
17       PAD C@ DUP MINUS C
18       (F) DEL ;
19
20 0 VARIABLE JX 0 VARIABLE JY
21
22 : V? JSTK
23        J0 2/ JX @ -
24        J1 2/ JY @ - OR
25      IF J0 2/ DUP JX !
26         J1 2/ DUP JY !
27         C/L * + !CUR V
28      ENDIF ;
29 DECIMAL ;S
30
31
```

```
SCR 24
 0 DECIMAL
 1 : DL 1 C 1 DEL ;
 2 : Q SCROL ." >" QUERY ;
 3 : J SCROL 0 JX ! 0 JY ! V
 4     BEGIN EDITOR ?TERMINAL DUP
 5     IF DUP
 6  ASCII G = IF GET     ENDIF DUP
 7  ASCII W = IF W       ENDIF DUP
 8  ASCII S = IF SP      ENDIF DUP
 9  ASCII X = IF X       ENDIF DUP
10  ASCII K = IF K       ENDIF DUP
11  ASCII H = IF H       ENDIF DUP
12  ASCII B = IF B       ENDIF DUP
13  ASCII N = IF N       ENDIF DUP
14  ASCII Q = IF DROP 3 ENDIF DUP
15  ASCII U = IF Q U     ENDIF DUP
16  ASCII P = IF Q P     ENDIF DUP
17  ASCII I = IF Q I     ENDIF DUP
18  ASCII R = IF Q R     ENDIF DUP
19  ASCII O = IF (I)     ENDIF DUP
20  ASCII D = IF Q D     ENDIF DUP
21        12 = IF DL     ENDIF DUP
22        BL = IF SCROL ." CMD>"
23                QUERY INTERPRET
24              ENDIF
25              SCR @ RANGE SCR ! V
26              #LOCATE JY ! JX !
27     ENDIF V?
28     03 = UNTIL ;
29 FORTH DEFINITIONS DECIMAL ;S
30
31
```

```
SCR 25
 0
 1
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
```