OSI-FORTH 2.0

FIG-FORTH 1.1

User Manual

Implemented By

R. A. Parise

&

M. DeGeorge

Distributed By

TECHNICAL PRODUCTS CO.

BOX 12983 University Station

Gainesville, Florida 32604

This publication has been made available by the FORTH Interest Group, P.O. Box 1105 San Carlos, CA 94070

DISK INFORMATION

As delivered, the FORTH system assumes that there are two types of disks. These are system disks and library disks. System disks are set up as follows:

TRACK 8"	TRACK 5 1/4"	CONTENTS
0-8	0-13	OS65DV3.2
9	14	BEXEC*
11-14	16-20	FTHSYS
15-76	21-39	BLOCK STORAGE

Tracks 15-76 (21-39)* are set up with twelve (eight) sectors of 256 bytes each. Since each FORTH screen is made up of four sectors, each track will hold three (2) complete FORTH screens. Starting with track 15 (21), these screens are numbered from 0 to 185 (37) on the system disk.

Library disks start with track 1 and have the same structure of twelve (eight) sectors per track. This results in 228 (78) screens on each disk. On a dual drive system, if drive 0 is selected (by typing DR0), then the first screen on drive 1 is 186 (38) and the screens continue through 413 (115). FORTH automatically switches drives for the user when any screen number greater then 185 (37) is requested. System disks may be accessed in Drive 1 by typing DR1 and using screen numbers between 0 and 185 (37).

With a single drive system, a library disk (normally in drive 1) may be accessed by typing 12 (8) OFFSET ! (CR) then requesting screens 0-227 (77). When you put the system disk back in drive 0, you must type DR0 (CR) before you can correctly access the screens.

*Numbers in parentheses refer to mini-floppy values.

FORTH VOCABULARY

The following words have been added to the basic FIG-FORTH vocabulary either for the sake of convenience or to bring the system up to the FORTH-79 standard.

J - Returns the index of the next outer loop to the parameter stack. Used only in nested DO loops.

I' - Returns the index upper limit of a DO loop to the parameter stack.

J' - Returns the index upper limit of the next outer DO loop to the parameter stack. Used only in nested DO loops.

n U. - Prints the integer n as an unsigned number.

n m U.R - Prints the integer n as an unsigned number right justified in a field of width m.

n ARRAY cccc- Creates an n element array in the dictionary with name cccc. Later execution of cccc leaves the address of the first element of the array on the stack.

DISK "string"- Allows OS-65D to execute the command string.

CLR - Clears the 540 video display memory.

BINARY - Switches the base to binary.

OCTAL - Switches the base to base 8.

n 0> - Returns a one to the stack if n is greater than zero. Otherwise, returns a zero.

NOT - This is an alias for 0=.

CODE cccc- Creates a dictionary entry named cccc and switches the context vocabulary to ASSEMBLER.

n ()DIM cccc- Creates an n element array in the dictionary with the name cccc. Later execution of m cccc leaves the address of the mth element of the array on the stack.

EXPANDING THE SYSTEM

As supplied, your FORTH system automatically LOADs screen 1 when it boots up. Screen 1 initially only contains a ;S in line zero. This provides the user with a very simple method of expanding the system which is LOADed on boot up. If, for example, you develop a number of words which are contained in screen 50, you may cause these words to be compiled on boot by simply editing a 50 LOAD before the ;S in screen 1. Another example would be the case in which you obtained a floating point package which began on screen 20. It would be convenient to have the number 20 defined as a constant named FLOATING. This would allow compilation of the floating point package to be initiated by typing FLOATING LOAD instead of remembering the screen number. This could be accomplished by editing 20 CONSTANT FLOATING directly into screen 1.

UTILITIES

The OSI-FORTH system is supplied with several utilities which provide copy, initialization, and reconfigure functions. These utility words are located in screens 6 through 12 on your disk. The following pages describe the use of these words.

BLOCK-INITIALIZATION ON NEW DISKS

There are four words provided in the screen labeled (BLOCK INITIALIZATION WORDS) for initializing the block storage area on new disks.

These words cover the following situations.

SYS0-INIT	-	Initializes a system type disk on drive 0
SYS1-INIT	-	Initializes a system type disk on drive 1
LIB0-INIT	-	Initializes a library type disk on drive 0
LIB1-INIT	-	Initializes a library type disk on drive 1

To initialize the block storage areas on disk, simply LOAD this screen and execute the appropriate word. These words write blank blocks on the entire block storage area! Make sure you have the correct disk in the drive before executing them.

SYSTEM RECONFIGURE

THE OSI-FORTH system, as supplied, is configured for 32K (24K for mini-floppy) of memory and four disk buffers (enough to hold one screen). (See Figure 1). Also, the backspace character is set to 'RUBOUT' as is standard in most FORTH systems.

The System Reconfigure screen allows modification of these parameters and automatic rewriting of the newly configured system onto the disk. To reconfigure your system, first LOAD the screen labeled (SYSTEM RECONFIGURE). The word BSPACE allows changing the backspace character in the following manner:

HEX 5F BSPACE	will rewrite the system boot-up parameters such that SHIFT-0 (the normal backspace character in OSI software) is the backspace.

To change the memory and buffer configuration to 48K and twelve buffers (enough to hold three screens simultaneously), for example, one would type:

HEX C000 OC CHANGE [illegible]

NOTE--DO NOT execute these words on your prime backup copy of the system. They rewrite part of the system and a disk error or user error could clobber the system! Only reconfigure working copies!

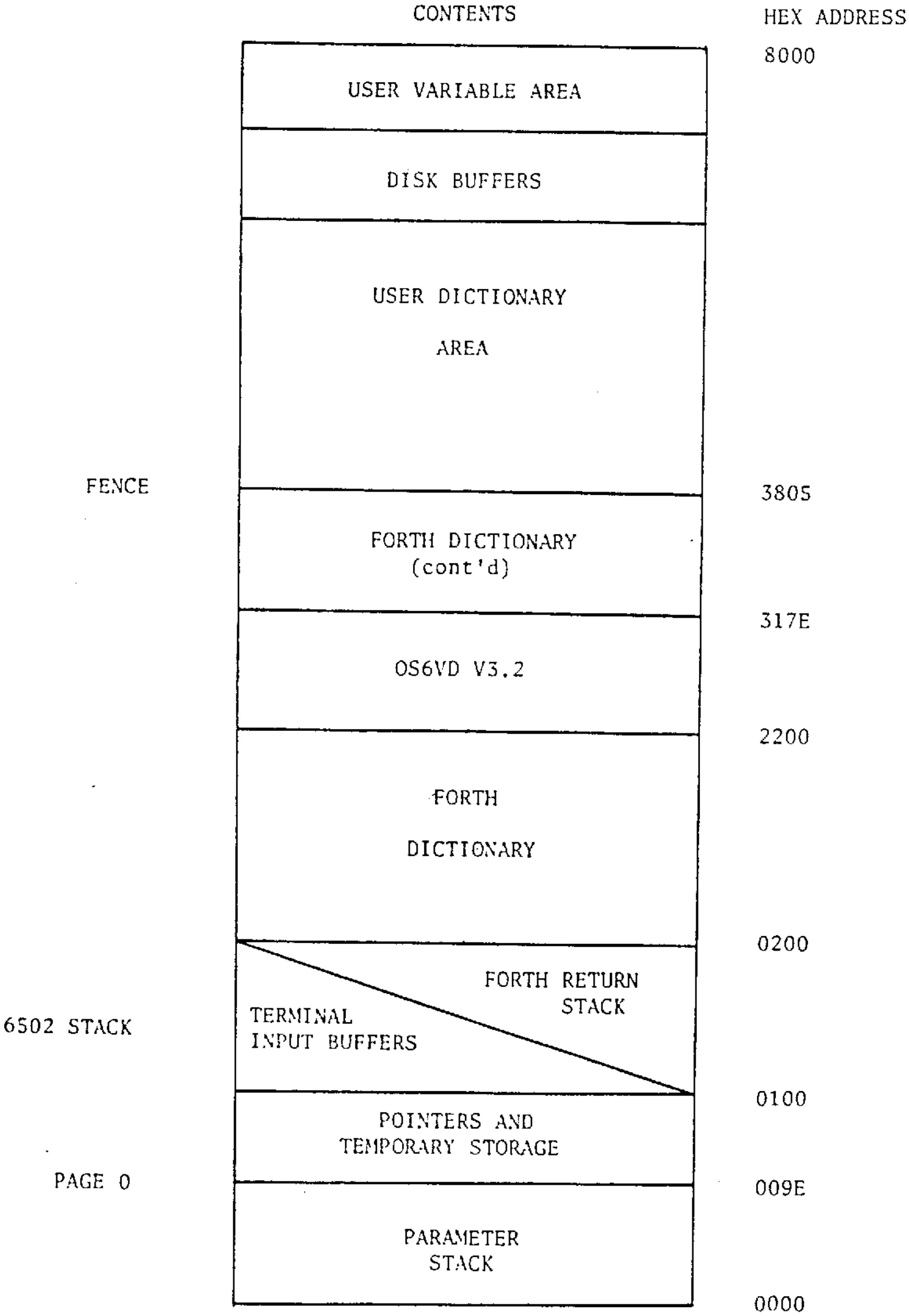


Fig. 1 FORTH System Memory Map

Creating a FORTH diskette for OSI C1P/MF with OS65D V3.2

---

1) Initialize a blank diskette with the normal method.

2) Copy tracks 0 to 14 from the FORTH master diskette to the 'new' diskette.

3) Boot the system with the FORTH master diskette.

4) Type " 8 LOAD " to load screen 8, insert the 'new' diskette and type " SYSCOPY ".

5) Insert the FORTH master diskette and type " 12 LOAD ".

6) Insert the 'new' diskette and type " SYSO-INIT ". This will take several minutes for completion.

7) Insert the FORTH master diskette and type " 9 LOAD ", then type " 0 14 COPY-SCREENS ", and follow the instructions on the screen.

8) When step 7 is complete the 'new' diskette should now be able to boot up FORTH...

SYSTEM COPY

System copy words are provided for both single and dual drive systems. The word SYSCOPY, when executed, will prompt the user for a fresh disk and then proceed to write the basic system onto the new disk. In order to produce a bootable disk, the following steps must also be taken:

1. The disk must contain OS65D V3.2 (including track zero)

2. FORTH's own BEXEC* must be transferred to the new disk.

3. The block storage area must be initialized using the appropriate utility word

4. Screen 1 must contain a ;S

5. Screens 4 and 5 should contain the error messages

COPYING FORTH SCREENS

There are two methods of copying screens in the OSI-FORTH system. To copy one screen from one location on the disk to another, simply use the word COPY (see EDITOR VOCABULARY). If, however, you wish to copy a number of screens from one disk to the same location on another disk, the screen copy utilities are provided. To use this utility, simply LOAD the appropriate screen (i.e., single or dual drive) and type the following;

```
n  m  COPY-SCREENS
```

This will initiate copying of screens n through m to drive 1 or will prompt the user to switch disks at the appropriate time in single drive systems.

?TERMINAL FOR SERIAL SYSTEMS

The FORTH word ?TERMINAL tests for a terminal key pressed (usually ESC) and returns a 1 to the stack if true , or a 0 if false. Since this word does not wait for input, it cannot request input through the OS-65D I/O distributor but must perform its own check. OSI-FORTH V2.0 is supplied such that ?TERMINAL specifically tests the OSI polled keyboard for ESC. If you intend to use your OSI-FORTH on a serial system, you may modify ?TERMINAL such that it tests device 1 for a key pressed. The screen labeled ?TERMINAL for serial systems will rewrite the ?TERMINAL code automatically for this purpose. This screen will execute while LOADing so do not LOAD it until you are sure you are ready.

Instructions

1. Make a copy of your original disk using the system copy utility.
2. Using your copy (never do this to your original), LOAD the screen labeled ?TERMINAL FOR SERIAL SYSTEMS and then recopy this disk onto itself (i.e., use the same disk for original and copy) using the system copy again.
3. ?TERMINAL will now respond to key closures (any key) on device 1.

THE EDITOR

When OSI Forth is finished loading, there are several sets of definitions in memory. These sets are called vocabularies. The purpose of having vocabularies is to enable faster searches by Forth since you may specify which vocabulary to start with. One of these vocabularies is the Editor. The EDITOR vocabulary may be listed by typing EDITOR VLIST and carriage return (CR). The other vocabularies may be listed by typing FORTH VLIST (CR) and ASSEMBLER VLIST (CR).

When listing a vocabulary, you will notice that, in addition to the words in the specified vocabulary, the Forth vocabulary is also listed. This is because all vocabularies are linked to Forth so that if a definition is not found in the context vocabulary, the Forth dictionary will also be searched.

Before getting on with actual examples of using the Editor, a few ground rules are in order. Forth source text is normally edited with disk storage areas called screens. Each screen contains 16 lines (0-15) with a possible 64 characters per line. This gives a total of 1K bytes of data which occupy four disk blocks in the Forth operating system.

Rule #1

The first line (line 0) is reserved for descriptive text of the screen contents. Line 0 is what is displayed when using the INDEX command. For example, 0 20 INDEX (CR) would cause line 0 of screens 0 through 20 to be typed to the display.

Rule #2

Don't crowd the screens. Group the material in small, logical units with plenty of comments. It's bad enough reading your own Forth definitions let alone trying to figure out what someone else is trying to do. There are 186 screens per disk. Lots of room.

Rule #3

Don't use line 15 for text if you can avoid it. Save it for the words ;S and -->. It makes things easier to follow.

As a matter of good practice, the following hints are offered:

1. Put the colon definition name, with a brief, descriptive comment, on one line with (comments and) the actual definition on the following line(s).
2. Try and keep definition small enough to fit on a few lines.
3. Don't put more than one definition on a line.
4. Leave two blanks between each word in a definition and try to set off other subunits (such as DO Loops) by three spaces on a separate line.
5. Don't put unreleated definitions on a screen. There is no penalty for empty space when the screen is loaded into memory.
6. Don't leave more than 3 consecutive blank lines anywhere in a screen or before the ;s or -->.

OK, now for an example. (Refer to the Editor Glossary for the full definitions of the commands.) Suppose you wish to edit screen 14. First, invoke the Editor by typing EDITOR (CR). Then 14 LIST (CR). You should get a list of screen 14 which looks like Fig. 2. You could also type 14 EDIT (CR). This would have done the same thing.

First, let's copy this screen to avoid losing it while learning to use the EDITOR. To copy it to screen 15 (an empty screen), use the copy command 14 15 COPY (CR). To be sure it was done properly, do 15 LIST or 15 EDIT. From now on, we will work with the copy on screen 15. Suppose you want to enter a line of text or code or overwrite an existing line.

```
 0   ( EXAMPLE EDITING SCREEN)
 1   VOCABULARY TEST ( DEFINE A NEW VOCABULARY NAMED TEST )
 2   DECIMAL TEST DEFINITIONS ( DECLARE BASE AND CURRENT VOCAB )
 3
 4   : 10COUNT  ( DEFINE A WORD TO COUNT TO 10 )
 5           10 0 DO CR I . LOOP ;
 6
 7   : HELLO ( PRINT HELLO UNTIL ESC IS PRESSED )
             BEGIN ."HELLO!" CR ?TERMINAL END ;
 8
 9
10   : ?BASE ( PRINT THE CURRENT BASE IN BASE 10 )
11           BASE @ DUP DECIMAL . BASE ! ;
12
13   ;S ( TRANSFER CONTROL BACK TO KEYBOARD )
14
15
```

Fig. 2 Forth Editor Example

Do this with the P command. For example, let's put a new title on line 0. Type the following: 0 P (FIRST TRY WITH EDITOR) (CR). Forth will respond with OK. If you type an L (CR), the screen will be relisted with the latest updates or insertions.

Now suppose we wish to change all occurrences of the number 10 in the text to 25. This is done with the following commands.

```
TOP      (set edit cursor to top of screen)

F 10     (find first occurrence of 10)

B        (move cursor to beginning of the 10)
         [Note:  you could have also used the -2 M command]

X 10     (delete the 10)

C 25     (insert 25)

N 10     (find next occurrence of 10)

B

X 10

C 25

N 10

B

X 10

C 25
```

Now list the screen by typing L (CR). You could have also done the same thing by using only the X and C commands. Do this to change the 25 to 15.

```
Ex.      TOP

         X 25

         C 15

         X 25

         C 15

         X 25

         C 15

         L
```

Now, what if you want to move a whole line of text somewhere else on the screen. For example, move line 0 to line 14. This can be done by the use of the line buffer PAD. Do the following:

```
         0 D     (delete line 0 and put it in PAD)

        14 R     (replaces line 14 with what is in PAD
                 (i.e., line 0))

         L       (relist new screen)
```

If you wanted to duplicate line 0 on line 14, you would have used 0 H instead of 0 D in the above example.

Once all editing on a screen has been completed, you must save it by typing FLUSH (CR). Forth only saves those blocks that have been altered since the screen has been listed. To illustrate this, list the original screen by typing 14 EDIT, then FLUSH. Forth will not bother resaving this screen, but will just respond with OK. Load the screen by 14 (or 15) LOAD CR. If all went well, a VLIST will show that the definitions on this screen have been added to the dictionary in RAM along with the vocabulary name TEST. All of the definitions on this screen are now available for use. Try HELLO (CR). This will type the word Hello! until the ESC key is pressed. If you no longer want or need the vocabulary TEST, you can remove it by typing FORGET TEST CR. Do a VLIST to see that it is no longer in the dictionary.

If you made a mistake in the screen such that it will not load and you try to load it again, you may find several copies of the words in the dictionary and a warning message telling you that some of the words are not unique. You should FORGET the bad definitions and fix the screen before trying to reload. To avoid writing random garbage on the disk, it is a good idea to do an EMPTY-BUFFERS command whenever you have a problem involving disk I/O. Also, any time an error message is generated, the base is reset to decimal and the context vocabulary is set back to Forth which means that if you were using the Editor, you must reset the context dictionary back to the Editor by typing EDITOR CR.

GLOSSARY OF EDITOR WORDS

n EDIT - Clears the 540 video display memory, switches the context vocabulary to EDITOR and lists screen n.

" text" - Places the text terminated by a " in the buffer PAD.

n P text - Puts text on line n.

n D - Deletes line n but keeps it in PAD.

n S - Moves lines n to 15 down one loosing line 15 and making line n empty. SPACES ONE

n E - Erases line n (fills with blanks)

n H - Transfers line n to PAD. HOLDS Line n

n M - Moves editing cursor the signed amount n.

n T - Type line n and place editing cursor at the beginning of the line. Also leavess the line in PAD.

L - Clears the 540 video memory and relists the current screen.

n R - Replace line n with the text in PAD.

n I - Insert text in PAD into line n.

TOP - Sets the pointers for the editing cursor to the top left of the screen.

n CLEAR - Erases screen n (fills with blanks). FLUSH must be executed after CLEAR.

FLUSH - Writes all updated blocks back to disk.

n m COPY - Copies screen n onto screen m.

F text - Finds first occurrence of text starting at the current cursor position. Types out the line and positions the editing cursor after text.

N - Finds the next occurrence of the text previously found by F.

B - Back up cursor to before text in PAD.

X text - Delete the next occurrence of the following text.

TILL - Deletes remainder of line beginning at the cursor position.

C text - Insert text starting at the cursor position.

THE ASSEMBLER

During boot up, Forth also loads a 6502 assembler. Most of the familiar 6502 assembler mnemonics are included with a few additions which make life easier. The only opcodes which are not implemented are LDX ZP,Y and STX ZP,Y. The biggest difference is in the use of the Forth Reverse Polish style notation. Also, all Forth assembler mnemonics end with a comma (i.e., LDA, ADC, etc.). A very important point to remember here is that Forth makes very heavy use of the X register. Because of this, a location XSAVE has been set aside for temporary storage of the X register whenever your programs make use of it.

In general, all assembly routines are preceded by the Forth word CODE which creates a dictionary entry for a named routine and switches the context vocabulary to the ASSEMBLER. At this point, an example will go a long way. Suppose for some obscure reason you wanted to load the accumulator with $41 and store it at location $D250 32 times using the X register as the counter. Traditionally, the code would appear as follows:

```
EXAMPLE   STX    XSAVE
          LDX    #$20
BEGIN     LDA    #$41
          STA    $D250
          DEX
          BNE    BEGIN
          LDX    XSAVE
          JMP    NEXT
```

Written with the FORTH assembler, the same routine would look like this:

```
1    Hex  Code  Example
2               XSAVE  STX,   20   #   LDX,
3                    Begin,
4                          41  #  LDA,  D250  STA,
5                          DEX,    0=   END,
6               XSAVE  LDX,  NEXT   JMP,
```

In this example, Line 1 sets the base to 16, enables the Assembler as the context vocabulary, and names the following code as EXAMPLE when it is loaded. Line 2 saves the X register and does an immediate load of X with Hex 20. Line 4 loads the accumulator with Hex 41 and stores it at D250. Line 5 decrements X and tests for equality with 0. If X is not equal to 0, the end statement causes a branch back to begin. If X does equal 0, then line 6 restores the original value of the X register and lets Forth return to the address interpreter (NEXT). The use of Begin, and End, allow you freedom from having to compute backward branches. The other Forth conditionals are listed in the Assembler Appendix.

To handle the forward branch condition, the FORTH assembler uses the IF,...ELSE,...THEN, structure which is similar to some Basics and Pascal. The ELSE, is optional in Forth as in many other languages. With this construct, a value on top of the stack is tested by one of the 6502 conditionals and if the result is non-zero, everything between IF, and ELSE, is executed and the program skips to THEN,. If the tested value is zero, the program skips to the code following ELSE, and executes everything up to THEN,.

All FORTH CODE definitions must end in a jump back to the FORTH interpreter. There are several entry points possible depending upon what action is desired with respect to the stack. These entry points have the following names:

NEXT — begins execution of the next word with no effect on the stack

PUSH — pushes the accumulator as LSB and the top of the 6502 stack as MSB onto the Forth stack as a 16-bit value before executing the next word

PUSHOA — pushes the accumulator as LSB and a zero as MSB onto the Forth stack before executing the next word

POP — removes the top value from the Forth stack before executing the next word

POPTWO — removes the top two values from the Forth stack

PUT — replaces the top value of the Forth stack with the accumulator and top of the 6502 stack

In addition to the standard 6502 assembler mnemonics, the following words are supplied as part of the assembler vocabulary.

BEGIN, - Labels the beginning of a loop structure in a CODE definition.

END, - Marks the end of a BEGIN--END loop in a CODE definition. Must be preceded by one of the conditional tests; 0=, CS, 0<, NOT
Actually assembles a branch back to BEGIN.

IF, - Assemble a conditional forward branch to THEN, or ELSE,. Must be preceded by one of the conditional tests 0=, CS, 0<,NOT .

ELSE, - First assembles an unconditional branch to be used by THEN, then computes the branch from IF, to HERE, and stores it back at the branch instruction previously assembled by IF,.

THEN, - End of IF,...ELSE,...THEN, construct. Computer forward branches to HERE from IF, OR ELSE, and stores the value back at the branch instruction previously assembled by IF, or ELSE,

0= - Precedes IF, or END,. Causes a branch on not equal to zero (BNE) to be assembled by IF, or END,.

CS - Precedes IF, or END,. Causes a branch on carry clear (BCC) to be assembled by IF, or END,.

0< - Precedes IF, or END,. Causes a branch on plus (BPL) to be assembled by IF, or END,.

NOT - Follows one of the previous three conditional tests to cause the inverse branch instruction to be assembled by IF, or END,.

All assembler mnemonics end with a comma (example: LDA, ADC, CLC, etc.)

UTILITY SCREENS FOR BOTH EIGHT-INCH AND MINI-FLOPPY SYSTEMS:

```
SCR # 0
  0
  1
  2
  3
  4
  5
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15

SCR # 1
  0 ;S
  1
  2
  3
  4
  5
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15

SCR # 2
  0
  1
  2
  3
  4
  5
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15
```

UTILITY SCREENS FOR BOTH EIGHT-INCH AND MINI-FLOPPY SYSTEMS:

```
SCR # 3
  0
  1
  2
  3
  4
  5
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15


SCR # 4
  0
  1 STACK EMPTY
  2 DICTIONARY FULL
  3
  4 ISN'T UNIQUE
  5
  6 DISK RANGE ?
  7 STACK FULL
  8 DISK ERROR !
  9 BLOCK NUMBER TOO BIG
 10 DEVICE NOT READY
 11
 12
 13
 14
 15        OSI-FORTH      TECHNICAL PRODUCTS CO.      NOV. 9, 1980

SCR # 5
  0 ( ERROR MESSAGES )
  1 COMPILATION ONLY, USE IN DEFINITIONS
  2 EXECUTION ONLY
  3 CONDITIONALS NOT PAIRED
  4 DEFINITION NOT FINISHED
  5 IN PROTECTED DICTIONARY
  6 USE ONLY WHEN LOADING
  7 OFF CURRENT EDITING SCREEN
  8 DECLARE VOCABULARY
  9
 10
 11
 12
 13
 14
 15
```

UTILITY SCREENS FOR MINI-FLOPPY SYSTEMS:

```
SCR # 6
  0 ( SYSTEM RECONFIGURE )
  1 VOCABULARY SYSTEM
  2 HEX SYSTEM DEFINITIONS
  3
  4 : BSPACE 20E ! DISK "SA 16.1=0200/8" ;
  5
  6 : CHANGE  ( SET NEW TOP OF MEMORY AND # OF DISK BUFFERS)
  7         >R 80 - DUP DUP R> 104 * - DUP DUP
  8         ' FIRST  !   USE !   PREV !
  9         ' LIMIT  !   210 !
 10         DISK "SA 16.1=0200/8"
 11         DISK "SA 18.1=1200/8" ;
 12
 13
 14
 15 DECIMAL  ;S


SCR # 7
  0 ( ?TERMINAL FOR SERIAL SYSTEMS )
  1 HEX
  2
  3 ( XQTER )
  4         SCR @ 4 * 3 + BLOCK ' ?TERMINAL
  5         2 - @ 10 CMOVE
  6
  7 DECIMAL FORTH    ;S
  8
  9
 10 ( THE FOLLOWING LINES CONTAIN NON-PRINTING  )
 11 ( CHARACTERS ... DO NOT MODIFY !!!!! )
 12 [illegible])[illegible]L[illegible]
 13
 14
 15


SCR # 8
  0 ( SINGLE DRIVE SYSTEM COPY  )
  1 VOCABULARY SYSTEM
  2 SYSTEM DEFINITIONS
  3 : SYSCOPY    ( COPIES BASIC FORTH SYSTEM ONTO NEW DISK )
  4          ." PLACE NEW DISK IN DRIVE AND HIT RTN"
  5          CR KEY DROP
  6          DISK "SA 16,1=0200/8"
  7          DISK "SA 17,1=0A00/8"
  8          DISK "SA 18,1=1200/8"
  9          DISK "SA 19,1=1A00/8"
 10          DISK "SA 20,1=327E/8"
 11          EMPTY-BUFFERS  ;   ;S
 12
 13
 14
 15
```

UTILITY SCREENS FOR MINI-FLOPPY SYSTEMS:

```
SCR # 9
( SINGLE DRIVE SCREEN COPY )
FORTH DEFINITIONS
LIMIT FIRST - B/BUF / CONSTANT NBUF

: WRITE NBUF 0 DO UPDATE PREV @ +BUF DROP
        PREV ! LOOP FLUSH ;

: READ NBUF 0 DO DUP I + BLOCK DROP LOOP ;

: COPY-SCREENS
        1+ B/SCR * SWAP B/SCR * DO
        ." PLACE SOURCE DISK IN DRIVE AND HIT RTN"
        CR KEY DROP I READ
        ." PLACE DESTINATION DISK IN DRIVE AND HIT RTN"
        CR KEY DROP WRITE NBUF +LOOP ;
;S
```

```
SCR # 10
( DUAL DRIVE SYSTEM COPY ETC. )
VOCABULARY SYSTEM
SYSTEM DEFINITIONS
: SYSCOPY    ( COPIES BASIC FORTH SYSTEM ONTO NEW DISK )
         ." PLACE INITIALIZED DISK IN DRIVE B AND HIT RTN"
         CR KEY DROP
         DISK "SELECT B"
         DISK "SA 16,1=0200/8"
         DISK "SA 17,1=0A00/8"
         DISK "SA 18,1=1200/8"
         DISK "SA 19,1=1A00/8"
         DISK "SA 20,1=327E/8"
         DISK "SELECT A"
         EMPTY-BUFFERS  ;    :S


```

```
SCR # 11
( DUAL DRIVE SCREEN COPY )

DECIMAL FORTH DEFINITIONS

( n,m ) : COPY-SCREENS   ( COPIES SCREENS n, TO m
                          TO DISK IN DRIVE B    )
               1+ 4 * SWAP 4 * DO I BLOCK I OFFSET @ +
               320 + 0 R/W LOOP ;


  ;S





```

UTILITY SCREENS FOR MINI-FLOPPY SYSTEMS:

```
SCR # 12
  ( BLOCK INITIALIZATION WORDS )
   FORTH DEFINITIONS

  : BLOCK-INIT FIRST 256 BLANKS SWAP
           DO FIRST I OFFSET @ + 0 R/W LOOP DR0 ;

  : SYS0-INIT  ( INITIALIZE SYSTEM DISK ON DRIVE 0 )
                 DR0 0 152 BLOCK-INIT ;
  : SYS1-INIT  ( INITIALIZE SYSTEM DISK ON DRIVE 1 )
                 DR1 0 152 BLOCK-INIT DR0 ;
  : LIB0-INIT  ( INITIALIZE LIBRARY DISK ON DRIVE 0 )
                 8 OFFSET ! 0 312 BLOCK-INIT DR0 ;
  : LIB1-INIT  ( INITIALIZE LIBRARY DISK ON DRIVE 1 )
                 DR0 328 ~~639~~ BLOCK-INIT ;
                         640
  ;S
```

```
SCR # 13
  ( TRACE, NOTRACE )
  HEX FORTH DEFINITIONS

  CODE TRACE   ( ENABLE FORTH TRACE FUNCTION )
           20 # LDA,   7DF STA,   C1 # LDA,
           7E0 STA,   02 # LDA,   7E1 STA,
           NEXT JMP,

  CODE NOTRACE  ( DISABLE FORTH TRACE FUNCTION )
           XSAVE STX,   3 # LDX,   EA # LDA,
           BEGIN,   7DE ,X STA,   DEX,   0= END,
           XSAVE LDX,   NEXT JMP,

   FORTH DECIMAL ;S


```

```
SCR # 14
  ( EXAMPLE EDITING SCREEN )
  VOCABULARY TEST ( DEFINE NEW VOCABULARY NAMED TEST )
  DECIMAL TEST DEFINITIONS ( DECLARE BASE AND CURRENT VOC. )

  : 10COUNT    ( DEFINE A WORD TO COUNT TO 10 )
          10 0 DO  CR I .  LOOP ;

  : HELLO  ( PRINT HELLO UNTIL 'ESC' IS PRESSED )
          BEGIN   ." HELLO!" CR  ?TERMINAL  END ;

  : ?BASE ( PRINT CURRENT BASE IN BASE 10 )
          BASE @ DUP  DECIMAL  .  BASE ! ;

  ;S (TRANSFER CONTROL BACK TO KEYBOARD )


```

BEXEC* FOR MINI-FLOPPY SYSTEMS

```
1 X=PEEK(10950):POKE 8993,X : POKE 8994,X
2 IF PEEK(57088)=223 THEN POKE 9794,37
5 PRINT:PRINT:PRINT
10 PRINT" OSI-FORTH    VERSION 2.0
20 PRINT
30 PRINT" TECHNICAL PRODUCTS CO.
40 PRINT" 1 FEB 1981"
50 PRINT
100 DISK!"IO 10,10"
110 DISK!"MEM 2E79,2E79"
120 PRINT#5,"EXIT";CHR$(13);
130 PRINT#5,"IO 10,80"CHR$(13);
135 PRINT#5,"CA 0200=16,1";CHR$(13);
136 PRINT#5,"CA 0A00=17,1";CHR$(13);
137 PRINT#5,"CA 1200=18,1";CHR$(13);
138 PRINT#5,"CA 1A00=19,1";CHR$(13);
139 PRINT#5,"CA 327E=20,1";CHR$(13);
142 PRINT#5,"GO 0200";CHR$(13);
145 PRINT#5,"EMPTY-BUFFERS";CHR$(13);
146 PRINT#5,"HEX 205 2A4F !";CHR$(13);
147 PRINT#5,"PAD DUP 8 EA FILL 2CDC 5 CMOVE";CHR$(13);
148 PRINT#5,"DECIMAL 1 LOAD";CHR$(13);
150 IFPEEK(10950)=2THENPRINT#5,"DISK IO 02,02";CHR$(13);:GOTO170
160 PRINT#5,"DISK IO 01,01";CHR$(13);
170 END
```

```
IST

 1 X=PEEK(10950):POKE 8993,X : POKE 8994,X
 2 IF PEEK(57088)=223 THEN POKE 9794,37
 3 POKE 9625,76:POKE 9626~~,80~~:POKE 9627,~~233~~
 4 POKE 61440,3:POKE 61440,16 2Ø8           251
 5 PRINT:PRINT:PRINT
 10 PRINT" OSI-FORTH    VERSION 2.0
 20 PRINT
 30 PRINT" TECHNICAL PRODUCTS CO.
 40 PRINT" 1 FEB 1981"
 50 PRINT
 100 DISK!"IO 10,10"
 110 DISK!"MEM 2E79,2E79"
 120 PRINT#5,"EXIT";CHR$(13);
 130 PRINT#5,"IO 10,80"CHR$(13);
 135 PRINT#5,"CA 0200=16,1";CHR$(13);
 136 PRINT#5,"CA 0A00=17,1";CHR$(13);
 137 PRINT#5,"CA 1200=18,1";CHR$(13);
 138 PRINT#5,"CA 1A00=19,1";CHR$(13);
 139 PRINT#5,"CA 327E=20,1";CHR$(13);
 142 PRINT#5,"GO 0200";CHR$(13);
 145 PRINT#5,"EMPTY-BUFFERS";CHR$(13);
 146 PRINT#5,"HEX 205 2A4F !";CHR$(13);
 147 PRINT#5,"PAD DUP 8 EA FILL 2CDC 5 CMOVE";CHR$(13);
 148 PRINT#5,"DECIMAL 1 LOAD";CHR$(13);
 150 IFPEEK(10950)=2THENPRINT#5,"DISK IO 02,02";CHR$(13);:GOTO170
 160 PRINT#5,"DISK IO 01,01";CHR$(13);
 170 END

Ok
```

UTILITY SCREENS FOR EIGHT-INCH DISK SYSTEMS:

```
SCR # 6
  0 ( SYSTEM RECONFIGURE )
  1 VOCABULARY SYSTEM
  2 HEX SYSTEM DEFINITIONS
  3
  4 : BSPACE 20E ! DISK "SA 11,1=0200/B" ;
  5
  6 : CHANGE  ( SET NEW TOP OF MEMORY AND # OF DISK BUFFERS)
  7          >R 80 - DUP DUP R> 104 * - DUP DUP
  8          ' FIRST  !   USE !   PREV !
  9          ' LIMIT  !   210 !
 10          DISK "SA 11,1=0200/B"
 11          DISK "SA 12,1=0D00/B" ;
 12
 13
 14
 15 DECIMAL  ;S


SCR # 7
  0 ( ?TERMINAL FOR SERIAL SYSTEMS )
  1 HEX
  2.
  3 ( XQTER )
  4          SCR @ 4 * 3 + BLOCK ' ?TERMINAL
  5          2 - @ 10 CMOVE
  6
  7 DECIMAL FORTH    ;S
  8
  9
 10 ( THE FOLLOWING LINES CONTAIN NON-PRINTING  )
 11 ( CHARACTERS ... DO NOT MODIFY !!!!! )
 12 ▌▙)▟▜L█
 13
 14
 15


SCR # 8
  0 ( SINGLE DRIVE SYSTEM COPY  )
  1 VOCABULARY SYSTEM
  2 SYSTEM DEFINITIONS
  3 : SYSCOPY    ( COPIES BASIC FORTH SYSTEM ONTO NEW DISK )
  4
  5           ." PUT NEW DISK IN DRIVE AND HIT 'RTN' "
  6           CR KEY DROP
  7           DISK "SA 11,1=0200/B"
  8           DISK "SA 12,1=0D00/B"
  9.          DISK "SA 13,1=1800/B"
 10           DISK "SA 14,1=317E/B"
 11           EMPTY-BUFFERS  ;
 12
 13 ;S
 14
 15
```

UTILITY SCREENS FOR EIGHT-INCH DISK SYSTEMS:

```
SCR # 9
( SINGLE DRIVE SCREEN COPY )
FORTH DEFINITIONS
LIMIT FIRST - B/BUF / CONSTANT NBUF

: WRITE NBUF 0 DO UPDATE PREV @ +BUF DROP
         PREV ! LOOP FLUSH ;

: READ NBUF 0 DO DUP I + BLOCK DROP LOOP ;

: COPY-SCREENS
         1+ B/SCR * SWAP B/SCR * DO
         ." PLACE SOURCE DISK IN DRIVE AND HIT RTN"
         CR KEY DROP I READ
         ." PLACE DESTINATION DISK IN DRIVE AND HIT RTN"
         CR KEY DROP WRITE NBUF +LOOP ;
;S
```

```
SCR # 10
( DUAL DRIVE SYSTEM COPY ETC. )
VOCABULARY SYSTEM
SYSTEM DEFINITIONS
: SYSCOPY     ( COPIES BASIC FORTH SYSTEM ONTO NEW DISK )
           ." PLACE INITIALIZED DISK IN DRIVE B AND HIT RTN"
           CR KEY DROP
           DISK "SELECT B"
           DISK "SA 11,1=0200/B"
           DISK "SA 12,1=0D00/B"
           DISK "SA 13,1=1800/B"
           DISK "SA 14,1=317E/B"
           DISK "SELECT A"
           EMPTY-BUFFERS  :
;S
```

```
SCR # 11
( DUAL DRIVE SCREEN COPY )

DECIMAL FORTH DEFINITIONS

( n.m ) : COPY-SCREENS    ( COPIES SCREENS n, TO m
                            TO DISK IN DRIVE B     )
                 1+ 4 * SWAP 4 * DO I BLOCK I OFFSET @ +
                 924 + 0 R/W LOOP ;


  ;S
```

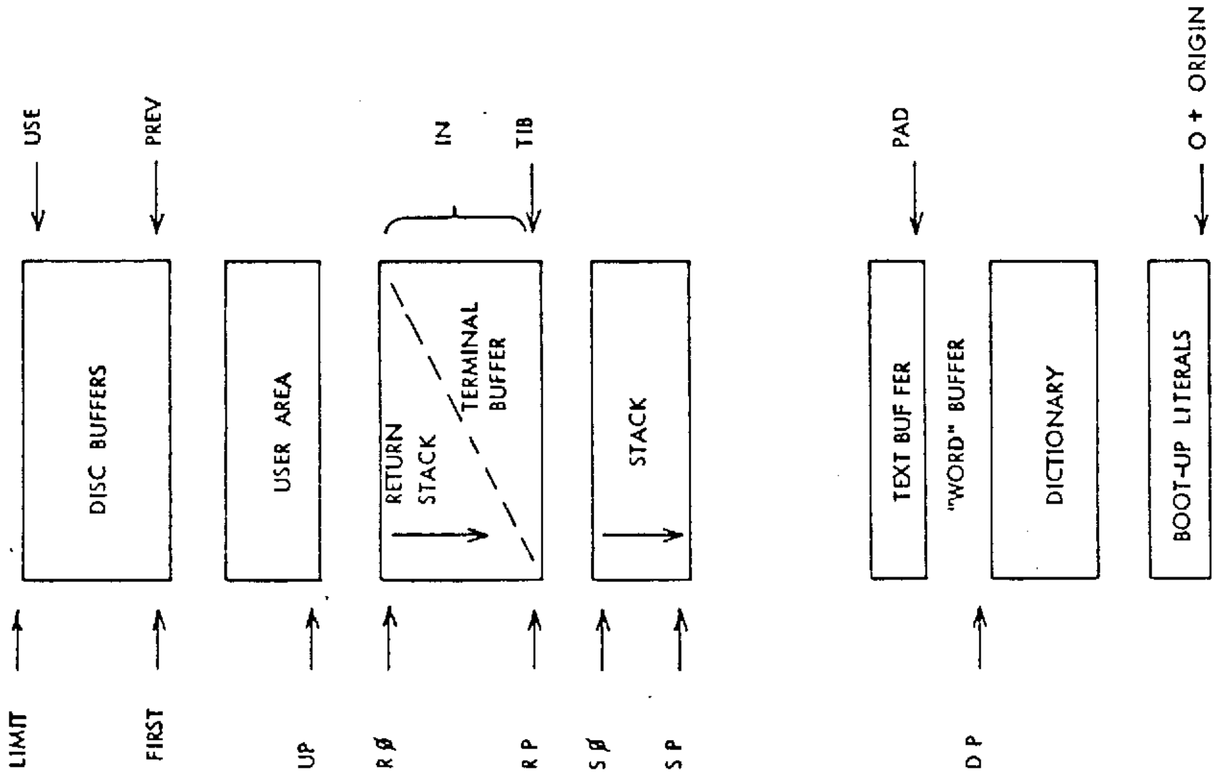
UTILITY SCREENS FOR EIGHT-INCH DISK SYSTEMS:

```
SCR # 12
( BLOCK INITIALIZATION WORDS )
 FORTH DEFINITIONS

: BLOCK-INIT FIRST 256 BLANKS SWAP
        DO FIRST I OFFSET @ + 0 R/W LOOP DR0 ;

: SYS0-INIT  ( INITIALIZE SYSTEM DISK ON DRIVE 0 )
             DR0 0 744 BLOCK-INIT ;
: SYS1-INIT  ( INITIALIZE SYSTEM DISK ON DRIVE 1 )
             DR1 0 744 BLOCK-INIT DR0 ;
: LIB0-INIT  ( INITIALIZE LIBRARY DISK ON DRIVE 0 )
             12 OFFSET ! 0 912 BLOCK-INIT DR0 ;
: LIB1-INIT  ( INITIALIZE LIBRARY DISK ON DRIVE 1 )
             DR0 936 1847 BLOCK-INIT ;

;S
```

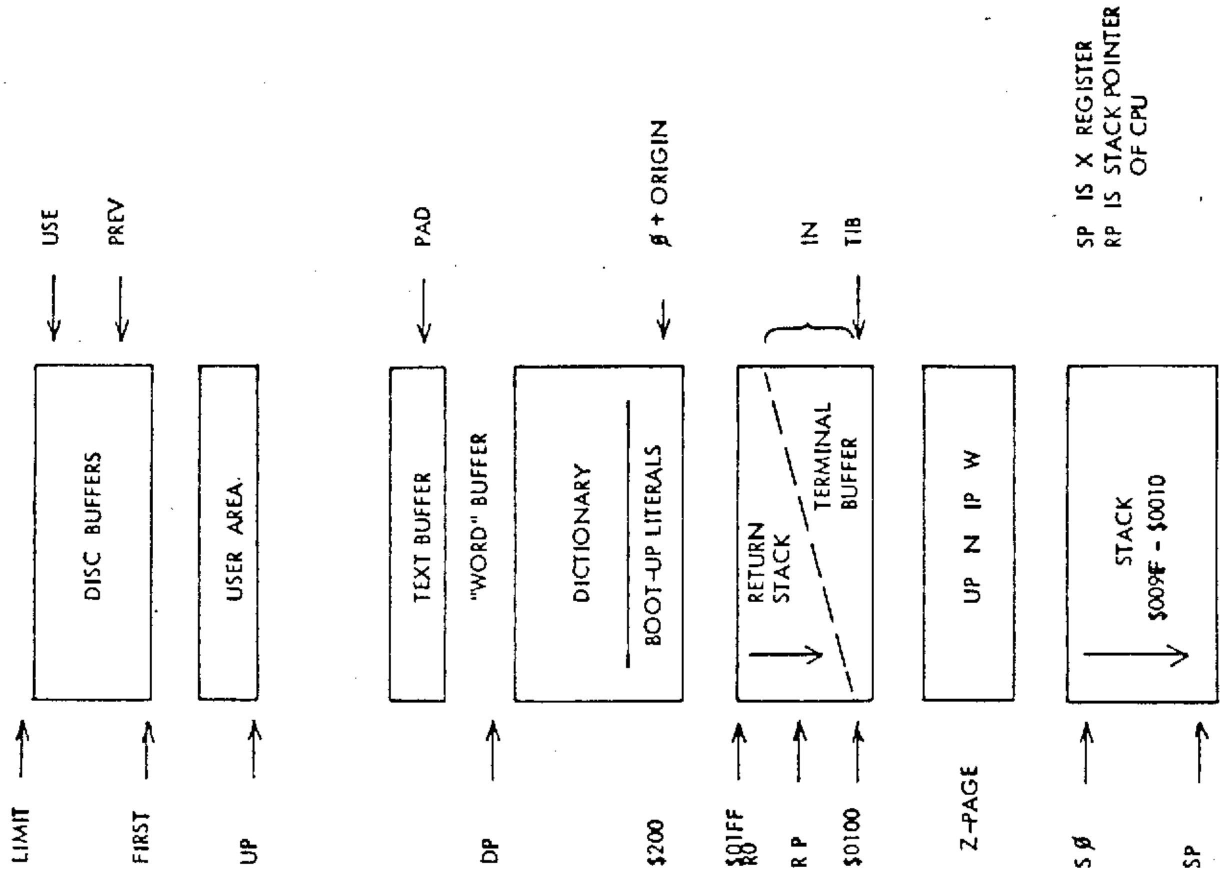
```
SCR # 13
( TRACE, NOTRACE )
HEX FORTH DEFINITIONS

CODE TRACE  ( ENABLE FORTH TRACE FUNCTION )
         20 # LDA,   7DF STA,   C1 # LDA,
         7E0 STA,   02 # LDA,   7E1 STA,
         NEXT JMP,

CODE NOTRACE  ( DISABLE FORTH TRACE FUNCTION )
         XSAVE STX,   3 # LDX,   EA # LDA,
         BEGIN,   7DE ,X STA,   DEX,   0= END,
         XSAVE LDX,   NEXT JMP,

 FORTH DECIMAL ;S

```

```
SCR # 14
( EXAMPLE EDITING SCREEN )
VOCABULARY TEST ( DEFINE NEW VOCABULARY NAMED TEST )
DECIMAL TEST DEFINITIONS ( DECLARE BASE AND CURRENT VOC. )

: 10COUNT   ( DEFINE A WORD TO COUNT TO 10 )
        10 0 DO  CR I .  LOOP ;

: HELLO  ( PRINT HELLO UNTIL 'ESC' IS PRESSED )
        BEGIN   ." HELLO!" CR  ?TERMINAL  END ;

: ?BASE ( PRINT CURRENT BASE IN BASE 10 )
        BASE @ DUP  DECIMAL  .  BASE ! ;

;S (TRANSFER CONTROL BACK TO KEYBOARD )

```

STANDARD
fig-FORTH MEMORY MAP



6502
fig-FORTH MEMORY MAP



BEXEC* FOR EIGHT-INCH DISK SYSTEMS

```
1 X=PEEK(10950):POKE 8993,X : POKE 8994,X
2 IF PEEK(57088)=223 THEN POKE 9794,37
5 PRINT:PRINT:PRINT
10 PRINT" OSI-FORTH    VERSION 2.0
20 PRINT
30 PRINT" TECHNICAL PRODUCTS CO.
40 PRINT" 1 FEB 1981"
50 PRINT
60 PA=62464:CA=PA+1:PB=PA+2:CB=PA+3
70 POKECA,0:POKECB,0:POKEPA,0:POKEPB,255
80 POKE CA,4:POKECB,4
100 DISK!"IO 10,10"
110 DISK!"MEM 2E79,2E79"
120 PRINT#5,"EXIT";CHR$(13);
130 PRINT#5,"IO 10,80"CHR$(13);
132 PRINT#5,"CA 0200=11,1";CHR$(13);
134 PRINT#5,"CA 0D00=12,1";CHR$(13);
136 PRINT#5,"CA 1800=13,1";CHR$(13);
138 PRINT#5,"CA 317E=14,1";CHR$(13);
140 PRINT#5,"GO 0200";CHR$(13);
145 PRINT#5,"EMPTY-BUFFERS";CHR$(13);
146 PRINT#5,"HEX 205 2A4F !":CHR$(13);
147 PRINT#5,"PAD DUP 8 EA FILL 2CDC 5 CMOVE":CHR$(13):
148 PRINT#5,"DECIMAL 1 LOAD":CHR$(13);
150 IFPEEK(10950)=2THENPRINT#5,"DISK IO 02,02";CHR$(13);:GOTO170
160 PRINT#5,"DISK IO 01,01";CHR$(13);
170 END
```

```
SCR # 66
( MISC  DEVICE CONTROL WORDS )
HEX FORTH DEFINITIONS

CODE DEVCHK  XSAVE STX.  265C LDA.  29C6 JSR.
        CS IF.  01 # LDA.  ELSE.  0 # LDA.
        THEN.  XSAVE STX.  PUSHOA JMP.

CODE HOMDRV  XSAVE STX.   2663 JSR.   XSAVE LDX.
        NEXT JMP.

: SETDRV  DUP 265C C@ = IF DROP DEVCHK ELSE 265C C!
        DEVCHK DUP 0= IF HOMDRV THEN THEN ;

: DR0 0B4 OFFSET ! ;    : DR1 39C OFFSET ! ;

DECIMAL ;S


SCR # 67
( BLOCK-WRITE )

HEX FORTH DEFINITIONS

CODE BLOCK-WRITE  ( WRITE ONE SECTOR TO DISK )
        EE LDA.       ( GET TRACK NUMBER )
        26BC JSR.     ( SET DRIVE TO TRACK )
        2754 JSR.     ( LOAD HEAD )
        27E1 JSR.     ( WRITE SECTOR )
        2761 JSR.     ( UNLOAD HEAD )
        NEXT JMP.     ( DONE )

 DECIMAL    ;S





SCR # 68
( BLOCK-READ )

HEX FORTH DEFINITIONS

CODE BLOCK-READ  ( READ ONE SECTOR FROM DISK )
       EE LDA.       ( GET TRACK NUMBER )
       26BC JSR.     ( SET DRIVE TO TRACK )
       2754 JSR.     ( LOAD HEAD )
       2967 JSR.     ( READ SECTOR )
       2761 JSR.     ( UNLOAD HEAD )
       NEXT JMP.     ( DONE )

DECIMAL    ;S




      OSI-FORTH      TECHNICAL PRODUCTS CO       NOV  9, 1980
OK
```

```
SCR # 69
  0 ( OSI DISK R/W )
  1
  2 HEX FORTH DEFINITIONS
  3
  4 : R/W SWAP DUP 237 > 9 ?ERROR DUP 39B > IF
  5        390 - 2 SETDRV ELSE 1 SETDRV THEN
  6        IF 1 SETDRV 1 A ?ERROR THEN
  7         C /MOD -BCD EE C! 1 + 265E
  8        C! SWAP FE ! 1 265F C! IF BLOCK-READ
  9        ELSE BLOCK-WRITE THEN ;
 10
 11 DECIMAL ;S
 12
 13
 14
 15


SCR # 70
  0 ( OSI DOS COMMAND INTERFACE )
  1 HEX FORTH DEFINITIONS
  2
  3 CODE DKEXEC  XSAVE STX.  2A84 JSR.  XSAVE LDX.
  4         NEXT JMP.
  5
  6 : DOSEXEC   E1 ! DKEXEC 2E1E E1 ! ;
  7
  8 : (DISK) R COUNT  1+ R> + >R  DOSEXEC ;
  9
 10 : DISK ?? STATE @ IF COMPILE (DISK) WORD HERE
 11        C@ 1+ ALLOT ELSE WORD HERE 1+ DOSEXEC THEN ;
 12   IMMEDIATE
 13 DECIMAL
 14     ;S
 15


SCR # 71
  0
  1
  2
  3
  4
  5
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15

       OSI-FORTH      TECHNICAL PRODUCTS CO       NOV  9. 1980
OK
```

```
SCR # 81
  0 ( J. ARRAY. ()DIM. ETC  )
  1 DECIMAL FORTH DEFINITIONS
  2
  3 CODE J  XSAVE STX.  TSX.  R 4 + LDA.  PHA.
  4       R 5 + LDA.  XSAVE LDX.  PUSH JMP.
  5
  6 : ARRAY <BUILDS 2 * HERE 2+ DUP , OVER ALLOT SWAP ERASE
  7         DOES> @  ;
  8
  9 : ()DIM <BUILDS 2 * HERE 2+ DUP , OVER ALLOT SWAP ERASE
 10         DOES> @ SWAP 2 * + ;
 11
 12 : OCTAL 8 BASE ! ;
 13 : BINARY 2 BASE ! ;
 14
 15 ;S


SCR # 82
  0 ( MORE USEFUL ADDITIONS )
  1 DECIMAL FORTH DEFINITIONS
  2
  3 : U R >R 0  <# #S #> R> OVER - SPACES TYPE ;
  4
  5 : U  0 U R SPACE ;
  6
  7 : NOT 0= ;
  8
  9 : 0> 0< 0= ;
 10
 11 : DUMP OVER + OVER  CR DO I 5 U R 8 0 DO
 12        I J + C@ 5 U R LOOP CR 8 +LOOP ;
 13
 14   ;S
 15


SCR # 83
  0 ( MORE ADDITIONS TO BASIC SYSTEM )
  1 HEX EDITOR DEFINITIONS
  2 : " 22 TEXT ;
  3
  4 DECIMAL FORTH DEFINITIONS
  5
  6 : MOVE 2 * CMOVE .
  7
  8 CODE I' XSAVE STX. TSX. R 2+ LDA. PHA. R 3 + LDA.
  9.          XSAVE LDX. PUSH JMP.
 10
 11 CODE J' XSAVE STX. TSX. R 6 + LDA. PHA. R 7 + LDA.
 12          XSAVE LDX. PUSH JMP.
 13
 14   ;S
 15

      OSI-FORTH      TECHNICAL PRODUCTS CO       NOV  9, 1980
OK
```

```
SCR # 114
 ( STRING EDITOR COMMANDS )
 : C  ( SPREAD AT CURSOR AND COPY IN FOLLOWING TEXT )
        1 TEXT PAD COUNT #LAG ROT OVER MIN >R
        FORTH R R# +! R - >R DUP HERE R CMOVE
        HERE #LEAD + R> CMOVE R> CMOVE UPDATE
        0 M ;

 FORTH DEFINITIONS DECIMAL

 ;S







SCR # 115
 ( ASSEMBLER )
 HEX ASSEMBLER DEFINITIONS
 0 VARIABLE MODE 4 ALLOT  23B CONSTANT PUSH
 242 CONSTANT NEXT 23D CONSTANT PUT 3F1 CONSTANT POP
 3EF CONSTANT POPTWO 307 CONSTANT SETUP AE CONSTANT IP
 5A6 CONSTANT BINARY EDF CONSTANT PUSHOA B1 CONSTANT W
 A6 CONSTANT N B5 CONSTANT XSAVE B3 CONSTANT UP
 :  .A -4 MODE ! -1 MODE 2+ ! ;
 : # -4 MODE ! 0 MODE 2+ ! ;
 : .X DUP ABS 100 < IF 8 MODE ! 0 MODE 2+ !
        ELSE 10 MODE ! 1 MODE 2+ ! THEN ;
 : .Y 0C MODE ! 1 MODE 2+ ! ;
 : X) -0C MODE ! 0 MODE 2+ ! ;
 : )Y 4 MODE ! 0 MODE 2+ ! ;
 : ) 20 MODE ! ;
 DECIMAL ;S


SCR # 116
 ( ASSEMBLER MNEMONICS ) HEX
 : MOP <BUILDS C, DOES> C@ MODE @ DUP 0= IF DROP
        SWAP DUP ABS 100 < IF SWAP 8 - C, C, ELSE SWAP
        C, , THEN ELSE + MODE 2+ @ DUP -1 = IF DROP
        C, ELSE IF C, , ELSE C, C, THEN THEN THEN
        0 MODE ! 0 MODE 2+ ! ;

 6D MOP ADC.   2D MOP AND.   CD MOP CMP,   4D MOP EOR.
 AD MOP LDA.   0D MOP ORA.   ED MOP SBC,   8D MOP STA.
 CE MOP DEC,   4E MOP LSR.   2C MOP BIT,   0E MOP ASL.
 EE MOP INC.   2E MOP ROL.   6E MOP ROR,   8E MOP STX.
 8C MOP STY,   AE MOP LDX,   EC MOP CPX,   CC MOP CPY,
 AC MOP LDY,
 : IMM MODE @ -4 = IF -C MODE ! THEN ;
 : LDX. IMM LDX, ; : CPX. IMM CPX. ; : CPY IMM CPY, ;
 : LDY. IMM LDY. ; DECIMAL   ;S

      OSI-FORTH      TECHNICAL PRODUCTS CO      NOV 9, 1980
OK
```

```
SCR # 117
( ASSEMBLER MNEMONICS )
HEX
: ABS <BUILDS C, DOES> C@ MODE @ DUP 20 = IF + C, ,
     ELSE DROP C, , THEN 0 MODE ! ;

4C ABS JMP,   20 ABS JSR,

: REL <BUILDS C, DOES> C@ C, C, ;

90 REL BCC,   B0 REL BCS,   F0 REL BEQ,   30 REL BMI,
D0 REL BNE,   10 REL BPL,   50 REL BVC,   70 REL BVS,

: BOT 0 ,X ;   : R 101 ,X ;   : SEC 2 ,X ;

DECIMAL  ;S


SCR # 118
( ASSEMBLER MNEMONICS )
HEX
: CPU <BUILDS C, DOES> C@ C, ;

00 CPU BRK,   18 CPU CLC,   D8 CPU CLD,   58 CPU CLI,
B8 CPU CLV,   CA CPU DEX,   88 CPU DEY,   E8 CPU INX,
C8 CPU INY,   EA CPU NOP,   48 CPU PHA,   08 CPU PHP,
68 CPU PLA,   28 CPU PLP,   40 CPU RTI,   60 CPU RTS,
38 CPU SEC,   F8 CPU SED,   78 CPU SEI,   AA CPU TAX,
A8 CPU TAY,   BA CPU TSX,   8A CPU TXA,   9A CPU TXS,
98 CPU TYA,

DECIMAL  ;S




SCR # 119
( ASSEMBLER CONDITIONALS )
HEX
90 CONSTANT CS  D0 CONSTANT 0=  10 CONSTANT 0<
30 CONSTANT 0>
: BEGIN, HERE ; IMMEDIATE

: END, C, HERE 1+ - C, ;

: IF, C, HERE 0 C, ;

: THEN, HERE OVER 1+ - SWAP C! ;

: ELSE, 18 C, 90 IF, SWAP THEN, ;

: NOT 20 + ;
 FORTH DECIMAL ;S
```

```
OK
```