fig-FORTH under OS-65U
Documentation
(PRELIMINARY)

Software Consultants
7053 Rose Trail
Memphis, TN 38134
(901) 377-3503

INTRODUCTION.

The Software Consultants fig-FORTH under OS-65U is a (mostly) faithfull implementation of the fig model as defined in the fig-FORTH Installation Manual & Glossary. The areas where we have deviated from the model are defined later in this document.

We assume that anyone who has purchased this package has at least some familiarity with the FORTH language, so we won't give a detailed explanation of the workings of the language. However, we would like to make a few statements to those who are not yet polished FORTH programmers.

Even more than some other languages, FORTH has a definate "learning curve" before you will feel confident in your programming. This learning curve will differ greatly among individuals, mostly depending upon the type of programming previously done.

Typically, assembler programmers will take to FORTH almost immediately, while those who's background is strictly BASIC require a longer period of time before they begin to "think" in FORTH. Regardless of your previous background, if you will take the time to become familiar with FORTH, we are sure you will agree that FORTH is a near perfect tool for writing most any type of application you can devise.

We strongly recommend that you join the Forth Interest Group, P.O. Box 1105, San Carlos, CA 94070. Current membership is $12.00/year in the US. Membership includes a subscription to Forth Dimensions. This magazine is currently the best avaiable forum for information concerning FORTH. Also available from fig are a number of manuals which can be of great help to the beginning FORTH programmer.

DEVIATIONS FROM THE MODEL.

This version of FORTH was designed with business applications in mind. Toward this end, several modifications of the original FORTH system have been made as defined below.

1. The words DR0, DR1 and OFFSET have been deleted. Since this version was designed mainly for hard disk based systems, they are not necessary.

2. The use of lower case in definitions is now allowed. As well as the additional flexability in naming, compile time is also increased by about ten percent.

3. An auto-load feature has been added which automatically loads screen three upon boot or when the words COLD or ABORT are executed. This was done to allow "turn-key" systems with minimal operator training.

4. A "quiet compile" feature has been added. If the variable QUIET is non-zero, the compiler will not report the "<NAME> ISN'T UNIQUE" message. This again is for turn-key systems.

5. The words LIST, INDEX, TRIAD, and ULIST have been removed from the source and written high-level. Normally they are loaded automatically, but may be eliminated from the auto-load screen to provide a "locked" system.

6. The word MON from the model has been replaced by the word BASIC, which will reload the BASIC interpreter and go to the console mode. CAUTION: Be sure and do a FLUSH before returning to BASIC to avoid losing any disk modifications.

7. All references to multiple sectors comprising a single FORTH screen have been removed since this is not required under OS-65U.

ENHANCEMENTS.

A number of useful routines have been included with the system. Each is described below. Please note that these routines were written by us and are NOT in the public domain.

TERMINAL & PRINTER TOOLS: In order to make dealing with the terminal and printer easier, a number of useful words have been defined on screens 6 thru 8. These words are well commented and should be self-explanatory.

LISTING WORDS: The words LIST, INDEX, TRIAD, and ULIST along with some supporting words are defined on screens 9 thru 11. Two types of modifications were made to the standard words. The first three words are made to operate differently depending on whether the output is to the terminal or printer. Also, the listing words have been modified to allow for object screens (defined below).

DOUBLE PRECISION SUPPORT: Screen 18 adds some words useful in dealing with double precision numbers.

DISK I/O SUPPORT: Screen 19 defines the word DISK which may be used to do direct disk access anywhere on the disk. This word may be used to read and write standard 65U disk files. For an example of it's use, see the OS-65U disk directory words.

Also on screen 19 is the constant TCB. This is the location of the transfer control block in 65U. You may change the current drive for disk access by storing the proper value at this address. I.E., to change to floppy drive B, do "1 TCB C!". All further disk access would then be from drive B.

CASE STATEMENT: Screen 20 contains a general case statement as found in such languages as Pascal. The syntax of the case statement is as follows:

```
    DO-CASES  ( start case. number to test on top of stack )
       n CASE ... ESAC  ( if n = tos then do everything between
          .                CASE and ESAC, then continue execution
          .                after CASES-DONE. This construct can
          .                be repeated any number of times. )
       default statements  ( any statements between the last CASE, ESAC
                             pair will be executed if none of the
                             previous cases was true. )
    CASES-DONE  ( end of case statment )
```

DISK DIRECTORY: Screens 21 thru 23 defines words which emulate the BASIC program "DIR". Once loaded, typing DIR will display the 65U disk directory. PRN DIR will send the directory to the printer. Note: Screens 18 and 19 must be loaded before loading screen 21.

FIND FILE IN DIRECTORY: Screen 48 defines the word FINDFL which will search the OS-65U directory for a given file name. The word TEST is also defined as an example of how to use FINDFL.

Before calling FINDFL, the name of the file to be found must be at HERE in the normal fashion left by the word WORD, i.e., the first byte is the length followed by the file name. If the file name is not found, the only entry on the stack after calling FINDFL will be the boolean flag 0 to denote failure. If the name is found, a 1 will be the first entry on the stack, followed by a double precision file length, and a double precision disk address. These entries may be used with the word DISK defined on screen 19 to do disk I/O on a standard OS-65U data file.

FINDFL requires the definitions on screen 21. Therefore, either the entire disk directory must be loaded prior to loading screen 48, or the contents of screen 21 only may be loaded.

RANDOM NUMBER GENERATOR: Screen 33 contains a simple random number routine which allows specifying the lowest and highest allowed numbers.

PRINT STACK CONTENTS: Screen 34 defines the word S? which will non-destructively display the contents of the stack in both decimal and hex. This is a very useful word during the debugging process.

LOAD/UNLOAD ASSEMBLER: Screen 35 contains an alternative way to use the standard fig assembler (on screens 12 - 17). Normally the fig assembler is loaded, then any required code words are defined. The only problem with this is that the assembler occupies about 1300 bytes of code, which are not used once the code words are compiled.

Screen 35 will load the assembler in high memory. After all required code words are defined, executing the word KILL.ASSM will remove the assembler. Therefore, code words may be used without the 1300 byte overhead which is normally required.

Note: At this time, there is still a "bug" in the load/unload assembler which will create problems if another vocabulary has been defined after screen 35 is loaded. Until this is fixed, we strongly suggest that all words defined (both code and high-level) between the time screen 35 is loaded and KILL.ASSM is executed should go in the FORTH vocabulary.

GET/PUT COMPILED CODE: The normal method of executing a FORTH application is to load the required colon definitions from disk and then execute them. The only difficulty is that loading definitions requires compiling them. While the FORTH compiler is fast, it is certainly not instantaneous. Therefore, it can take many seconds to load even a relatively small application.

For development purposes, the speed of the LOAD is not a problem, but for turn-key systems, execution of an application should be as fast as possible.

The definitions on screens 36 - 38 allow storing compiled code on disk and then recalling them for use later at extreme high speed. This is done by using the word PUT to write the memory image of compiled words to disk, then using the word GET to recall them.

The word PUT is used as <screen #> PUT <name>, where screen # is the first screen the compiled code will occupy, and name is the name of the first word in the dictionary to be put to disk. All the following words up to HERE are saved. Please note that variables and constants can be put to disk just as colon defined words can.

The word GET is used as <screen #> GET, where screen # is the first screen containing previously put code. A major word of caution: The dictionary MUST be in exactly the same condition as when the corresponding PUT was executed. If it is not, you can expect the system to just go totally out to lunch.

This is not a limitation, but a requirement for doing the GET in the fashion we are. GET is not a relocating linking loader, but a simple disk to memory routine. While we could have written GET to allow loading code onto a modified dictionary, the extra processing required would probably have required as much time (or even more!) as the normal LOAD command.

The words LOAD, EDIT, LIST, and INDEX have been modified to work correctly with object code screens. Attempting to LOAD or EDIT an object screen will give an error. LISTing an object screen will show the starting and ending memory addresses and length of the PUT block.

TERMINAL ORIENTED EDITOR: We've saved the best for last. If you have ever used the standard fig editor, you know it leaves a great deal to be desired. Screens 24 thru 32 contains a very flexible, easy to use, FORTH screen editor.

The normal method for editing a screen is by the command <screen #> EDIT. This will clear the screen, list the screen in the upper portion of the display, and position the cursor at the beginning of line zero. A number of control functions are allowed to simplify entry and editing of screen data.

1) The cursor control keys; up, down, left, and right; may be used to put the cursor anywhere on the display within the screen. All four functions "wrap around", i.e., pressing the up arrow at the top line will move the cursor to the same column on the bottom line. Likewise, pressing the left arrow key at the beginning of a line will move the cursor to the end of the previous line.

2) The tab key will move the cursor to the right by the number of spaces contained in the variable TABAMT. Shift tab will move the cursor to the left by the same amount. TABAMT is initially loaded with eight, but may be modified to any value you desire. The tab function also will wrap around.

3) CTRL E will erase the line containing the cursor.

4) CTRL S will insert a blank line at the position of the cursor. Line 15 will be lost.

5) CTRL D will delete the line containing the cursor and move all following lines up. Line 15 will be blank.

6) CTRL A will insert a single blank at the current cursor position. All remaining characters on the line will be moved to the right., The last character on the line is lost.

7) The delete key will remove the character presently under the cursor. All remaining characters on the line will be moved to the left. The last character on the line will be a space.

8) ~~CTRL P~~ CTRL N and CTRL O work in conjunction with each other. CTRL ~~P~~ N will save the line containing the cursor at PAD. CTRL O will replace the line containing the cursor with the line previously saved with CTRL ~~P~~N. This allows moving a line from one location to another on a screen, or moving a line from one screen to another.

9) ~~CTRL T~~ CTRL B will position the cursor to the beginning of line zero.

10) The escape key will end editing of the current screen. If any modifications were made, the screen will be marked as updated.

Also available within the EDITOR vocabulary are several other helpful words. The word N will edit the next screen. The word S will re-edit the last screen edited. The word P will edit the previous screen. <screen #> CLEAR will fill the given screen with blanks. <from screen> <to screen> COPY will copy the from screen onto the to screen.

If a compile-time error occurs, the word WHERE may be used to edit the screen containing the error and place the cursor at the location of the error.

INSTALLATION PROCEDURE.

As with any purchased software, the first thing you should do is copy this disk, and put the original away. The original disk should never be modified.

If you will be using our FORTH on a hard disk, you will need to create three files on the disk. The two files FORTH and BASIC should be created at the same size (or larger) as shown on the floppy. The file SCREEN may be created as large as you wish (max size is 64 Megabytes!). This is the file which contains the FORTH disk "work space". Each one K of disk space will give you one FORTH screen.

The file FORTH is a BASIC program which is run to initiate FORTH. It may be loaded from the floppy and saved to the hard disk. The file BASIC is a work file to hold the BASIC interpreter while FORTH is executing. It need not be copied. The file SCREEN may be moved using the OSI utility COPYFI.

If a larger SCREEN file is created, the new screens should be cleared before using them. Assume you create a one megabyte file for SCREEN (1024 screens). The following word should be defined and executed before attempting to use the new screens:

```
: CLEAN   EDITOR   1025   225   DO   I   CLEAR   LOOP   ;
```

If you will be using FORTH on a floppy based system, the following procedure should be used to prepare a working disk. Place an OS-65U disk in drive A and a blank disk in drive B. Using the COPIER utility, initialize the disk in drive B and copy the operating system onto it from drive A. Replace the disk in drive A with the delivered FORTH disk, and copy the files from drive A to B.

REQUIRED MODIFICATIONS.

The only modifications that will normally be required before you can take advantage of all the existing FORTH definitions are in the terminal and printer area. All terminal oriented commands are delivered set up for a Micro-Term, Inc. ACT-5A. If you are using another type of terminal, certain changes must be made.

Since the terminal oriented editor will not function until these changes are made, a minimal line oriented editor is provided on screens 45 and 46. After loading this, a screen may be modifyed by typing EDITOR, then LISTing the required screen, and using the following commands:

L : The L command will relist the current screen and show any changes which have been made.
<line #> P <text> : The P command is the method used to put text on a screen. The entered text will replace the existing line's data. Note that at least one space must follow the text, but the first space will not be put on the screen.
<line #> S : This is the same as the CTRL S in the video editor
<line #> E : Same as the video's CTRL E.
<line #> D : Same as the video's CTRL D.
<line #> H : Hold the designated line at PAD.
<line #> R : Replace the line with the data held at PAD.
<line #> I : Insert the data at PAD at the line entered and move following lines down. Line 15 is lost.

The words CLEAR and COPY are also provided. They are the same as those under the video editor.

The only changes that should be required are on screens 6 thru 8 and 28 thru 30. If you are not using a serial printer (BASIC device #3), then the value of the constant PRNVAL on screen 6 should be changed. The proper value for a parallel printer would be 10. For a device #8 serial printer the value would be 80. The 0C EMIT on line 4 of screen 7 should be changed to whatever is required to clear the screen on your terminal. Likewise, the 14 EMIT on line 6 of screen 8 should be set to the cursor positioning command. If your terminal requires the column address before the row address, then put a SWAP before the last two EMITs on this line.

The possible changes on screens 28 thru 30 are in the terminal oriented editor. In particular, the codes for the up, down, left, and right arrow keys may require changing. Note that the codes for each key is in decimal.

Once all changes have been made, executing "5 LOAD" will recompile all definitions loaded at boot and save the compiled code. Executing the word "COLD" will then make the changes effective.

RUNTIME MODIFICATIONS.

Quite often it is better to have several smaller FORTH disk work spaces rather than one large one. This can easily be done with this version of FORTH. The base disk address of the file used by FORTH (default is SCREEN) is contained at memory address zero. To change to another file for the work space, just store the proper disk address here.

The disk address is a double precision number, i.e., 4 bytes. If the double precision tools are loaded, the D! may be used to put the proper value at zero. If not, the disk address will need to be broken into two pieces and stored at zero (low order) and two (high order).

Changing from one floppy disk drive to another only requires storing the proper value at address 26A1 hex. If the disk I/O support tools are loaded, this value is stored in the constant TCB (transfer control block). This value is a single byte; 0 - 4 for drives A - D. It may be set with the C! word.

If either of the above changes are made, you should precede them with a FLUSH to write any updated buffers to disk, and an EMPTY-BUFFERS to prevent copying over any existing buffers to the new disk area.

Please note that this FORTH will accept screen numbers outside of the legal range for the file used as the disk work space. This was done to allow accessing other 65U disk files using the standard FORTH words, such as BUFFER and BLOCK.

IN CONCLUSION....

As with all our products, we at Software Consultants fully support our FORTH. If you have any questions or problems, do not hesitate to contact us. We would prefer that problems be communicated by mail, so that you may provide as much detail as possible in defining the problem, and we may be as explicit as possible with our answer. However, unlike some vendors (who shall remain nameless), we will gladly speak to you by phone if you have an immediate need.

As usual, the source code for FORTH is available on disk for our standard $10.00 nominal fee. Due to the large size of the source, it cannot conveniently be assembled with the standard OSI assembler. The source is delivered set up for assembly using the Pegasus Software A/65 assembler.

Unlike most interpreters or compilers, the FORTH source is mostly in FORTH itself. Modifications are normally much easier to make than it would be for a more common language. If you have any assembler experience, and feel that certain things could be done more to your liking, jump in! It really is quite a lot of fun!

While this FORTH is complete as delivered, and even includes a number of "extras", it is by no means the end of the line. We intend to offer several additions in the near future. String handling and floating point routines are currently working in house, and will be released shortly. Even more exciting developments are on the drawing board, so watch for our ads.