CHAPTER 2. BROWSING F83 SYSTEM

I assume that you have either followed the instructions as described in the README.TXT file on the original disk you obtained from Henry Laxen or Mike Perry and expanded the compressed files to the full length files comprising the F83 system, or somebody did the expansion for you and you have a set of floppy disks ready to be used to explore this interesting and powerful Forth operating system and language. If you did not have an expanded system, please read the instructions in the README file and then run the executable file RUNME. You will be guided step by step to create a set of disks which will contain all the files to be used by the F83 system.

In this chapter, I would like to show you what are contained in the F83 system and also the files on the disks and help you to get familiar with this system. All the commands and exercises mentioned in this chapter can be used freely to exercise the system so that you will gain certain degree of confidence to use them later when you will do programming. These commands will in no way disturb the information stored on the disks. The best way to learn them is to type them in on the keyboard and observe the results on the CRT terminal.

F83 system is very large comparing to earlier public domain Forth system like figForth. It has about 1000 words or instructions in its dictionary. However, most words are defined to support other high level words and are seldom used for ordinary programming purposes. Only a very limited number of words are used often and these are words that a user must learn and be fluent in them to use Forth productively. Included in this set of words are the required word set defined in the Forth-83 Standards, which is a minimum set of words allowing you to compose solutions to a wide range of programming problems, and the set of utility words in this F83 system which allows you to use the specific resources provided by your computer. I further assume that you have already had some knowledge on Forth by reading some textbook like Leo Brodie's 'Starting Forth', or its equivalent, and used a Forth system from some other source. Therefore, I will not try to explain in details the elementary functions and words common to most Forth system and only discuss those words unique in the F83 system. The purpose is to get you to know this system well enough so that you will be able to use it as a basis to build your application or your new Forth system.

In this chapter, all the words or instructions discussed are non-destructive. They will allow you to browse through the entire system and explore its riches without writing anything to any of the files. You must try them all and get to know them well before we get to the next chapter where we will try to edit files and make permanent changes on disks. However, it is recommended that you make some backup copies of the disks with the expanded files and only use the copies for the exercises, just to be safe.

2.1. LISTING THE WORD NAMES

Words or instructions in Forth are very powerful constructs. They have the essence of subroutines in FORTRAN, procedures in PASCAL and PL/I, characters in APL, macro instructions in assembly, and command files in operating systems. Because they are resident in a dictionary in the RAM memory of a computer, they are available for immediate execution or for compilation into other high level words. Words in the dictionary are arranged in the form of a linked list so that the execution procedure associated with a particular word can be located quickly by the Forth operating system. A very useful

utility word is defined to go through this linked list and print the names of all the words in the dictionary. It is called WORDS in Forth-83 Standard, a remarkable improvement over the old computerese name VLIST in the figForth model. Typing:

Figure 2.1. IBM PC-DOS files in F83 system

WORDS

on your keyboard will generate a long list of word names on the terminal, as shown in the following figures. On the list of Forth words, you will find all the regular Forth words for arithmetic operations like + , - , * , / , and other division and ratioing operators; the stack operators like DUP , DROP , SWAP , OVER and ROT; the memory operators like @ , ! , C@ , and C! ; etc. In fact, all the Forth-83 standard words are included in this list some where.

WORDS has few equivalent in other language or operating system. The Forth computer can tell you all the words it knows and which are available for your use, any time you care to browse. In other language or operating system, you have to go look them up in thick manuals and can never be sure that they are really in your system. WORDS reveals the current state of the dictionary. If you add more words to the dictionary, they will appear at the top of the name list. It is very handy when you are extending the system by defining new words and add them to the dictionary. In this case you will be interested in the words on the top of the dictionary and not the rest of the long listing. You can stop the name listing by pressing any key on the keyboard.

Fig. 2.2 shows you how to list the words in the FORTH vocabulary.

2.2. VOCABUALRY

The dictionary in Forth is usually not a single linked list of words or instructions, but contains a number of logically independent linked lists of words called vocabularies. The purpose of the vocabulary is three-fold: to shorten the time needed to search through the dictionary, to group functionally related words together, and to allow different words to share the same name. There are nine vocabularies defined in the F83 system. The names of these vocabularies can be displayed by typing the following word:

VOCS

and nine vocabulary names will be displayed on the terminal. The function and contents of these vocabularies are summarized in Table 2.1.

TABLE 2.1. VOCABULARIES IN F83

NAME	FUNCTIONS
ROOT	Words to assign vocabulary searching order. All vocabularies must be defined in this vocabulary.
FORTH	The main trunk vocabulary for all standard and system words.
EDITOR	All editing commands.
ASSEMBLER	All words needed to define low level machine code routines.
DOS	Words to use the underlying DOS utility.
USER	Words to define user variables.
SHADOW	Words to support shadow screens for comments and documentation.

BUG	Words to support F83 debugger.
HIDDEN	Miscellaneous supporting words not useful to the user.

Executing a vocabulary name makes the specified vocabulary the 'context' vocabulary. The system will search the context vocabulary first to locate a word entered by the user. The word WORDS displays only the list of words in the context vocabulary. Since normally the context

vocabulary is the FORTH vocabulary, executing WORDS usually displays the word names in the FORTH vocabulary as shown in Fig. 2.2. Executing WORDS after a vocabulary name will list the word names in that vocabulary, as shown in the examples in Figures 2.3-4.

Figure 2.2 FORTH words

Figure 2.3 ASSEMBLER and DOS words

Figure 2.4 Words in other vocabularies

The vocabulary structure in F83 is significantly improved as compared with the vocabulary structure in the figForth Model. It is more flexible in that the user can dynamically change the vocabulary searching sequence and specify up to to eight different vocabularies in the searching sequence. The speed of dictionary searching is also much faster than that in the figForth Model, because all vocabularies in the dictionary are hashed into four threads. In order to locate a word, only a quarter of a vocabulary needs to be scanned. This hashed searching greatly improves the speed of text interpretation and program compilation.

Two words are used to manage the vocabulary searching sequence: ONLY and ALSO. ONLY initializes the searching sequence and makes ROOT as the only vocabulary available for searching. In the ROOT vocabulary, all the other vocabulary names must be defined so that they are accessible. After ONLY is executed, executing any other vocabulary word will make that vocabulary the context vocabulary which becomes the first vocabulary to be searched during text interpretation. Executing ALSO pushes the context vocabulary on the top of a vocabulary stack and makes it the first resident vocabulary. Other resident vocabularies already in the vocabulary stack are pushed down so that they will be searched in order after searches in the context and the first resident vocabularies failed to locate a word.

The context vocabulary, for all practical purposes, is the equivalent to the context vocabulary in fig-FORTH. The resident vocabularies are extensions of the context vocabulary to allow the user to specify the number and the order of vocabularies to be searched in runtime.

To arrange the searching order as DOS-EDITOR-ASSEMBLER-FORTH, one has to execute the following command sequence:

```
ONLY  FORTH ALSO  ASSEMBLER ALSO  EDITOR ALSO  DOS
```

Here DOS becomes the context vocabulary and EDITOR is the first resident vocabulary to be searched. If a word cannot be located in either DOS or EDITOR, the ASSEMBLER and the FORTH vocabularies will be searched in turn.

Another word ORDER will list the context and the resident vocabularies on the terminal. It is a useful command to assure yourself the context environment you are in at any time. If you executed the above string of vocabulary words, typing

```
ORDER
```

results in the following display on the terminal:

```
Context: DOS EDITOR ASSEMBLER FORTH ROOT
Current: FORTH ok
```

indicating the desired vocabulary search order. The current vocabulary, in this case FORTH, is the vocabulary to which new word definitions are added. It will be discussed later.

The command WORDS behaves similarly to VLIST in figFORTH. However, WORDS only lists the names of words in the context vocabulary; therefore, WORDS must be preceded by the name of the vocabulary you wish to examine, like FORTH WORDS, DOS WORDS, etc. Since the list of names always starts with the word defined last, it is often used to see which word was compiled last. If there

were any error during disk file loading, you can find quickly where compilation stopped.

2.3. VIEWING SOURCE CODE OF WORD DEFINITIONS

Since there are so many words in the F83 system, it is impossible for anybody to remember the meaning and the function of all these words. Although the compiled object code of a word in the dictionary contains all the information about this word, it is not readily usable to casual users. F83 system provides a very interesting and powerful tool which permits the user to see the source code of any word in the system. This magic word is named 'VIEW'. If you wanted to see how the word LIST was defined, you should type:

```
VIEW  LIST
```

and the F83 system will open the file in which LIST was defined and display the screen containing the definition of LIST. On the top of the displayed screen, you will also find the name of the file. This is shown in Fig. 2.5.

To use the viewing facility, you must have all the source files on disks and have them properly inserted into appropriate disk drives. For some computers, the files fit on a single floppy disk. This is the ideal case because you don't have to worry about where a particular file is. For computers with smaller disk drives, the files must be spread over two or more drives. If the required file is not on the disk of your current disk drive, you have to log on to the drive where the file is located and repeat the viewing command, or insert the proper disk in the the log-on drive and repeat the viewing command.

Words are grouped by their functions and by the order of compilation into six major files in the F83 system:

METAnn.BLK	The meta-compiler.
KERNELnn.BLK	The trunk Forth system. Nucleus, interpreter and compiler.
EXTENDnn.BLK	Vocabulary and file words.
CPUnnnn.BLK	Assembler and CPU dependent words.
UTILITY.BLK	Editor, debugger, decompiler, printing and other utility.
HUFFMAN.BLK	Huffman compression.

where nn or nnnn identifies the CPU for which the F83 system is hosted. 80 for 8080 and Z80, 86 for 8086 and 8088, and 68 for 68000.

If you have to choose which files to put on a disk for viewing, I suggest that you put UTILITY.BLK, KERNELnn.BLK, and EXTENDnn.BLK on one disk and use it for viewing, because they comprise the majority of useful words that you might be interested in browsing.

F83 also comes with a built-in decompiler which can regenerate the source code from the object code in the dictionary. The decompiler word is 'SEE', followed by the name of the word you want to decompile. For example:

```
SEE  LIST
```

will display the sequence of words which define the function of LIST on the terminal. The displayed sequence of words does not match exactly the sequence in the original source code, because the control structures are not decompiled but simply represented by the corresponding runtime routines.

Nevertheless, the decompiled sequence does reveal the composition of the source code faithfully. The advantage of the decompiler over the viewing facility is that the decompiler is always available for you to browse words, even without the disk files.

The result of SEE LIST is shown in Fig. 2.5, at the bottom.

Figure 2.5 VIEW and SEE

2.4. SHADOW SCREEN DOCUMENTATION

Since most people think that a FORTH screen of 1024 bytes is too small to put inline documentation with the code in the same screen, the shadow screen technique was developed to give the user an extra screen to write comments and documentation for each source screen. This documentation screen is the shadow of the source screen.

F83 divides a screen file into two equal parts: the first half will be used for source code and the second half for documentation. One can toggle between a source screen and its shadow screen with the commands A and L. After viewing the source code in a source screen, the user can type A L and switch to the shadow screen to see the comments and documentation. Documentation thus provided in the F83 system is quite extensive, and you are encouraged to examine the shadow screens with their respective source screens. The shadow screens generally bring out the purpose and over-all function of words which are not obvious in the source definition.

2.5. FILES IN F83

F83 uses MS-DOS or CP/M operating system to access the terminal and the disk files. Using a readily

available operating system to host the F83 system has the advantage that it can be transported to a large number of computers with that operating system. It also allows the partitioning of the F83 system into several named files which are easier to handle than a simply blocked disk. Within a file, however, F83 system still deals with program or data in the 1024 byte block format as required by the Forth-83 standard. Most of the elementary file functions are

defined as Forth words. However, only a few high level words are needed by the user to use files to store and to retrieve programs and data.

Three simple Forth words have functions similiar to their DOS or CP/M counterparts: DIR lists on the terminal all the files on the current disk drive, A: makes drive A the current drive, and B: makes drive B the current drive. All file activities are processed for files on the current drive.

All the Forth words using the disk mass storage, such as BLOCK, BUFFER, FLUSH, etc., access the current file on the current disk. A file becomes the current file when it is opened by the command OPEN <filename>, and subsequent disk commands are directed to this file. In our previous example of the word VIEW, which displays the screen containing the source code in a file, the word VIEW actually opens the file containing the word definition and displays the requested block on the terminal. If you want to examine or to modify data or source in a specific file, you have to open it explicitly. Once a file is opened, you can display any block within that file.

The size of a file is usually specified when the file was created. The size in number of 1024 byte blocks can be recalled by the word CAPACITY. Execute CAPACITY and the number of blocks in the current file is returned on the stack. Source code files in the F83 system with the extension BLK are arranged to have the source code in the first half of the file and the shadow documentation in the second half.

To examine the contents of a BLK file, you can use the command INDEX to display the first lines in a range of screens. For example, to display the first lines of all the source code screens, you can type:

```
0  CAPACITY 2/  INDEX
```

and the first lines of those screens will be displayed on the terminal. By convention, the first line in a screen should always be a comment to the contents of this screen. Thus INDEX gives us the information equivalent to a directory in a file. An example of the index listing of the UTILITY.BLK file is shown in Fig. 2.6.

If you identify any screen of your interest, you can examine the detailed contents of this screen by the command LIST, preceded by the screen number:

```
1 LIST
```

will display the first text screen in a file. In all the F83 source code files, screen one is the load screen of the file, i. e., it contains commands that will load or compile the source screens in the rest of the file. There are also some comments in screen one indicating the packaging of the screens in the file.

To display the shadow documentation of any source screen, you should type:

```
A  L
```

The command A uses CAPACITY to calculate the screen number of the associated shadow screen, then makes it the current screen. The command L displays the current screen. Executing A and L again will display the source screen again.

Figure 2.6 Files and directory commands

2.6. PRINTING UTILITY

To make hard copy of the source screens and shadow screens, a simple method is to let the printer follow the terminal display. In the CP/M systems you can type the control P code on the keyboard to turn on the printer. Any character hereafter displayed on the terminal will also be printed. Now you can use any of the listing commands discussed in the last section to print index of a file or individual screens. However, you do not have control over the printing format. F83 provides some utility commands to print source code and shadow screens. If you have an EPSON printer capable of printing in condensed format, you can print the source screens side by side with their shadows on single 8.5" by 11" paper, which is very convenient when studying the source code.

The print utility allows you to print a range of screens on a printer. It must be properly initialized for your printer. If you do have an EPSON printer you have to initialize it by the following commands:

```
' EPSON  IS  INIT-PR
```

which initializes the vectored word, INIT-PR. The print format is 6 screen to a page with two 3 screen columns. The printer must be able to print 132 characters per line to fit two screens side by side on one page. The command to print a range of screens is SHOW:

```
1 30 SHOW
```

will print screens 1 to 30.

There are two versions of SHOW in F83. The version in FORTH prints 6 screens per page and the version in the SHADOW vocabulary prints 3 screens of source with their corresponding shadow screens:

```
1 30 SHADOW SHOW
```

prints source screens 1 to 30, 3 screens to a page with 3 shadow screens.

If your printer cannot handle 132 columns per line, you will have to use the command TRIAD to print three screens on a page.

To obtain the complete listing of a file in the source-shadow format, there is a simple command LISTING. LISTING was used to generate all the source listings as distributed with the F83 systems, with file name, page number, and footing. .

2.7. DEBUGGER

The debugger is designed to let the user single-step through the execution sequence of a high level definition. To invoke the debugger to trace a word, issue the following command:

```
DEBUG  <name>
```

where <name> is the word to be debugged. Nothing happens at this point. DEBUG sets things up so

that when the word is executed you will get a single step trace showing the word within <name> that is about to be executed and the contents of the parameter stack.

During the single stepping through a word, the name of the next word to be executed and the contents of the parameter stack are displayed on the CRT terminal. The debugger then waits for a key stroke on the terminal keyboard. Any key will cause the next word to be executed and the debugging information displayed. Three special keys, C, F, and Q, have the following functions:

Q Quit the debugging process and restore the debugged word to its original state for normal execution.

C Turn off the single stepping mechanism and let execution run to completion.

F Temporarily return to Forth system so that you can execute other Forth commands, for example, to change the data stack items. You must type RESUME to come back and continue the debugging process.

An example to single step through the execution of 1 LIST is shown in Fig. 2.7. Typing Q at the bottom of the list terminates the execution.

Figure 2.7 Debugging LIST