PART 1. INTRODUCTION TO F83 SYSTEM

CHAPTER 1. THE HERITAGE OF F83 SYSTEM

1.1. THE ROOTS OF THE F83 SYSTEM

Forth was invented by Charles Moore in the 60's as he developed specialized programming tools for various software projects and crystallized them into a language and operating system. Forth spread to many continents following the radio telescopes originally programmed by Mr. Moore when he was with the National Radio Astronomy Observatory. Forth was so prevalent in the astronomy communities that the International Astronomic Union formally adopted Forth as their standard programming language in 1974. The Forth familty tree is shown in Figure 1.1.

Mr. Moore and some of his colleagues left NRAO and formed Forth, Inc. to market Forth systems and services in 1972. Over the years, a series of Forth implementations was produced for commercial minicomputers and microcomputers. These products evolved into polyForth, which contained many advanced features such as interrupt drive I/O, multitasking-multiprogramming, database management, transcendental functions, and meta- or target compilation. It remains the most comprehensive Forth system in the Forth market place.

Forth users in Europe organized a user's group, called European Forth users Group (EFUG). To encourage the exchange of Forth programs and information, EFUG published a list of Forth words with standard definitions, commonly known as the Forth-77 Standard. It documented the then-often-used Forth words in an effort to prevent these words from mutation as more Forth systems were installed.

The Forth Interest Group was organized in 1978 to encourage the use of Forth on small personal computers. Two major activities sponsored by FIG in 1978 were the publication of the figForth Model and the organizing of the Forth Standards Team. Because of the low costs of the figForth source listings and the quality of these figForth implementations, fig-Forth became the de facto standard of Forth on small computers. The product from the Forth Standards Team, the Forth-78 Standard, however, was not as successful. It was soon orphaned by FIG. The Forth Standards Team went back to the drawing board and produced the Forth-79 Standard which was much more precise in wording and consistent in the naming of standard words. Many vendors including Forth, Inc. made genuine efforts to adopt this standard into their products.

Several problems kept the Forth Standards Team working. Among them, the more serious ones are the state dependence of many words, the loop structure, the representation of falsehood, integer division with negative divisor, and the naming of many words. These problems were resolved in the publication of the Forth-83 Standard in early 1984. The exhausted Forth Standards Team decided that no new Forth standard will be considered in the near future to let Forth-83 Standard have some time to establish itself in the Forth community.

Figure 1.1 The Forth family tree

1.2. ADVANCEMENTS IN FORTH-83 STANDARDS

Major improvements in Forth-83 Standards over previous Forth standards are briefly discussed here. Exhaustive discussions have appeared in Forth Dimensions, authored by the Secretary of the Forth Standards Team, Dr. Robert L. Smith. Some of the more significant features in the Forth-83 Standard are summarized here.

MONO-ADDRESSING

Four addresses are used to address different fields in a word definition in the dictionary: the name field address, the link field address, the code field address, and the parameter field address. To allow maximum implementation flexibility and code portability, the 83-Standard uses only one address, the compilation address. It is equivalent to the code field address in the fig-Forth model. The compilation address is the one returned by ' and FIND, compiled to colon definitions, and used by EXECUTE to run the word definition. Only one extra word is provided in the standard to access information stored in the parameter field: >BODY.

The importance of the compilation address cannot be overstressed. Mono-addressing recognized this characteristics of Forth. It is also beneficial that the compilation address serves as the focal point in locating information stored in the word definitions.

ELIMINATING STATE SMART WORDS

Many Forth words in the early standards execute differently depending upon whether the system is in the execution mode or compiling mode, like LITERAL, ' (tick), ." , etc. In Forth-83 the state smart words are either eliminated or separate words are defined for interpreting and compiling states, making the system less ambiguous and faster.

Old ' is split into two words: ' and [']. Old ." is split into .(and ." .

IMPROVED DO-LOOP

The DO-LOOP structure went through a major overhaul in Forth-83. The range of index is extended to 64K so that full memory range can be addressed in the loops.

LEAVE is made to terminate the loop immediately upon its execution rather than waiting until LOOP is executed.

IMPROVED DIVISION

Division is now floored towards negative infinity instead of rounded towards zero. It is more useful in that the quotient and modulus have a smooth change between positive-nteger and negative-integer domain when the divisor, the dividend, or both are negative.

REPRESENTATION OF THE TRUE FLAG

A true flag generated by the Forth-83 system is represented by -1 instead of 1 in the older standards. -1

is more useful than 1 in doing bit-wise logic operation.

Consequently, NOT can now be defined as one's complement operation instead of being simply an alias of 0=.

ZERO-BASED PICK AND ROLL

The top of the data stack is treated as the base of a memory area and addressing into this area is zero based like addressing regular memory areas.

WORD RETURNING AN ADDRESS

The word buffer was generally assumed to be at the top of the dictionary. However, this is implementation dependent. With WORD returning the address of the word buffer, the word buffer can be assigned to other memory areas. Practical usage of WORD always requires the address of the WORD buffer. Including the word buffer address in the WORD function is a convenience to the user. A slight speed advantage can also be realized.

1.3. CREATORS OF THE F83 SYSTEM

F83 is a very extensive language and operating system created by Henry Laxen and Mike Perry, two professional Forth programmers in Berkely, California. Both of them have been active in the Forth interest Group since its beginning, and participated in the work of the Forth Standard Team to develop the Forth-79 Standard and the Forth-83 Standard. They have published papers and written tutorials on Forth in Forth Dimensions and in the FORML (Forth Modification Laboratory) proceedings. As the Forth-83 Standard evolved, they felt the need of a complete Forth system based on this standard to carry the standard to Forth users and community, in the same way the figForth implementations on the 6 popular microprocessors carried the figForth model. The F83 system was the result of their efforts.

The F83 system was designed to use the CP/M operating system as its host for terminal input-output and disk interface, so it is rather straightforward to transport it to a variety of microcomputers using the CP/M system. It has been implemented for 8080-Z80, 8086-8088, and 68000 CPU's. Laxen and Perry put this system in the public domain, according to the tradition of the Forth Interest Group, as a vehicle to distribute the newly established Forth-83 Standard. Wil Baden in Los Angles ported it to the Apple II computer and named the system F83X.

F83 Version 1.0 was released by Perry and Laxen in September, 1983, shortly after the Forth-83 Standard was published. They also organized an F83 working group in the Northern California to evaluate the F83 system for practical applications. The F83 system, over the period of about a year, was enhanced and upgraded several times. The latest version, Version 2.1 was released in the summer of 1984. The authors promised that this version will not be modified in the near future, and it should be stablized for the user to get familiar with it and to be distributed to a wider audience.

This book is meant to be a reference manual to the F83 system. It was originally written for the F83 system Version 1.0 for the 8086 processor on a CP/M86 operating system. Since the release of Version 2.1, it was also upgraded to this version. Due to the overwhelming spread of the IBM Personal Computers and its compatible models, it was also modified for using with the F83 system for the MS-DOS system. As for the other F83 systems, it is useful as a reference for all the high level Forth commands. For low level machine code commands, the user will have to refer to the source code and documentation coming with the specific F83 system.

1.4. FEATURES OF F83 SYSTEM

F83 is not a toy language like most other public domain and some commercial Forth system. It contains all the necessary utilities and tools for the user to develop application programs conveniently and efficiently. Both executable object codes and the source codes are provided and distributed on floppy diskettes in the machine readable form. Although Laxen and Perry do not intend to provide support and consultation on the F83 system, as they called their publishing firm No Visible Support, Inc., the systems they distributed are of excellent quality and can stand on their own strength. Extensive documentation are provided in the form of shadow screens and in-line comments.

Laxen and Perry intended that F83 should demonstrate and bring out the best features in Forth as a professional programming language. Many utilities and tools were included in this system which are not generally available even in the best of the commercial Forth systems. Some of these utilities are listed here:

Editor
Assembler for the host CPU
Full BIOS/BDOS interface
Multiple file accessing
Four-way threaded dictionary
Dynamically defined vocabulary search order
Source code viewing
Decompiler
Debugger
Memory dumping
Multitasking
Shadow-screen documentation
Source and documentation printing
Forward referencing
Metacompiler
Huffman code compression and expansion

In realizing all these functions, F83 has more than 1000 words in its dictionary, comparing with about 300 words in the fig-Forth model and 130 words in the required word set in Forth 83-Standard. Casual users may not need to know the details of all these words. However, this whole F83 system is a huge reservoir of Forth programming examples from which serious Forth programmers can study and find ready solutions to many of their programming problems.

FigForth became the de facto standard of Forth on small computer systems because of the quality and the availability of the source listings distributed by the Forth Interest Group. The fig-Forth system is complete in the sense that it includes all the necessary system functions so that it can be implemented on real life microprocessors. The Forth 79-Standard, on the other hand, had not attained the popularity of fig-Forth in spite of the intensive lobbying efforts within and without the Forth Interest Group. The principal reason is that 79-Standard has to be supported by a system to be useful. Forth Interest Group expected that the support to 79-STandard would come from vendors of commercial Forth systems, and it did not provide executable systems in supporting the standard.

Forth-83 Standard is a refinement on the Forth-79 Standard. Many ambiguities in the Forth-79 Standard

were resolved and all required words are defined in better precision. The DO-LOOP structure was overhauled. However, Forth Interest Group maintained the policy to let Forth vendors to provide the system support to the Forth-83 Standard. Because of the reservation they had in the capability of vendor supports, Laxen and Perry built the F83 system as the bearer of the

Forth-83 Standard. They also realized that the Forth community has matured over the years and a minimal system like figForth will not satisfy the needs of Forth users in building applications and systems. To be the standard bearer, F83 had to go beyond the figForth model and provides the user with a complete program developing environment.

Figure 1.2 The standard bearer