C-Style Structures in ANS Forth

Traditional Forth is quite different than the C programming language. The inventor of Forth, Charles Moore, developed his language to allow programmers to have more freedom and control over their programming environment than with languages with rigid syntax. Forth predated C and became a popular language in the early days of personal computers. The 1980 issue of Byte magazine focusing on Forth was the largest selling issue in its history. But today there is a new generation of programmers who have been raised on C and C based operating systems many of whom have never heard of Forth or who have only heard rumors about what Forth was like thirty years ago.

One of the most common problems today for people trying to understand the Forth programming language is that they do not know the history of Forth and the logic behind its design or the methodology of its use. They try to put Forth in a C frame and find it lacking in that context. They either abandon it or use its extensibility to add to the language the features that they are familiar with in C to let them use it in a way that they are used to. I briefly addressed one of the issues in my Essential Forth essay where I tried to make the point that Forth was designed to provide a set of words in its kernel to allow the language to be extended into something resembling fourth generation database languages and with the most intuitive semantics and most efficient implementation of persistent database records.

Part of my motivation was the often asked question of how one implements C-style structures in Forth. This seems to be one of the first questions that C programmers ask about Forth. Mr. Moore's reply is often "I wouldn't." By that he means that Forth has other mechanisms for implementing data structures and the code that operates on them than the style used in C.

Other programmers answer the question by providing an example of Forth code to allow C-style structure use in Forth. This has been seen by many as one of the things that Forth has needed to be more popular with programmers familiar with C. But first Forth had to address portability in the C sense of the word, that the same source could be compiled on a wide range of different computer architectures.

Early Forths reached for portability by having a layer of architectural specific code and higher layers of more portable high-level Forth code. Forth usually provided a system-dependent assembler in a given implementation of the language to integrate low-level assembler definitions into the language as Forth words that are accessible in high-level Forth. The language implementations were usually 16-bit, meaning that Forth words were represented as 16-bit entities and items on the Forth parameter and return stack were 16-bit wide. This provided a 16-bit virtual machine that was very simple and easy to implement on almost any computer. Forth became the first language implemented on most new computers because of the ease of porting a Forth to new hardware. As an interactive language it was often used to debug new hardware and portability was defined in Forth by the ease of implementation of a 16-bit virtual machine. This made it very easy to bring a wide variety of primitive 8-bit computers up to the level of a 16-bit virtual machine.

As hardware moved to 16-bit and 32-bit and 64-bit designs people adapted their Forth virtual machine implementations to 32-bit or 64-bit stack and memory items. Double precision computations became less common than on the early 16-bit designs. On Aug 3, 1986 work began to develop an ANS Forth standard. It began with the F-83 model but added the important concept of differentiating address units from word sized cells. In 16-bit Forths on 8-bit addressing machines most programmers just used the Forth word **2*** to convert from a count of 16-bit cells to a byte address. ANS Forth introduced the word **CELLS** to avoid the assumption that Forth was 16-bit or that the architecture used byte addressing. On a 16-bit Forth on a byte addressing machine **CELLS** was defined as **2*** but on a 32-bit word size implementation on a byte addressing machine it was defined as **4*** and on a word addressing machine of any size it is defined as **1***.

ANS Forth introduced the words **CELLS** and **CHARS** to provide a portable mechanism that would allow ANS Forth code to be portable on a wide range of implementations. The stated purpose of ANS Forth in (Section 1.1) is:

"The purpose of this Standard is to promote the portability of Forth programs for use on a wide variety of computing systems, to facilitate the communication of programs, programming techniques, and ideas among Forth programmers, and to serve as a basis for the future evolution of the Forth language."

The standard also provided a portable optional files wordset to allow access to files implemented either in Forth or the files provided by external operating systems. The C language makes a clear distinction between the language keywords and syntax and what is external. The language includes no I/O inherently, but relies on library files to provide things like I/O. There are no C compilers that do not require libraries. C was developed to write an Operating System with the integral concept that everything should be treated as a file.

ANS Forth addressed a concern that many programmers had about Forth, that it lacked portable application libraries. The ANS Forth standard made it possible to write such libraries for Forth for the first time. This was seen as a great thing by a great many people, particularly those who were developing the standard and using it to market their products.

At the time I was a strong proponent of this new ANS Standard for Forth. I had seen the importance of standards in the industry to reduce programming effort and reduce risks in management of projects. The development of an official ANS Standard for Forth was also seen by many as a needed boost for the credibility of Forth. But I did have some concern that I saw a great difference in the way programmers approached different languages, particularly with regard to how many programmers had become completely dependent on standardized libraries.

One of the first portable ANS Forth application libraries to become a collective project among Forth programmers was the Forth Scientific Library Project. As I watched it evolve I began to become concerned about its direction. The many functions that it provided were great, but I saw many compromises made in order to make the code portable on a wide range of 16-bit, 32-bit, and 64-bit implementations of ANS Forth. Dr. Everett (Skip) Carter did a great job or organizing the effort and contributed a number of programs to the library himself as well as testing code on a variety of PC Forths, workstation Forths, and even on Cray. The effort was applauded by Elizabeth Rather, who chaired the ANS Forth Technical Committee, Ms. Rather wrote in comp.lang.forth on 10/01/1998

"What Forth desperately needs is more application-oriented libraries, but it seems that all programmers with time to spare want to do is implement yet another Forth. The Forth Scientific Library is a shining example of what we need more of!

The Forth Interest Group had installed a new board of directors that include Ms. Rather, Dr. Carter, myself, and others. Dr. Carter had been administering the FIG website where he put his examples of CGI scripting in ANS Forth, Forths written in C, and the Forth Scientific Library as the most prominent information presented. I had run for the board on the platform that I would make an effort to get FIG to publicize more information about what the inventor of Forth, Charles Moore, had done with Forth in the previous thirteen years since leaving Forth Inc. I was concerned that FIG had almost no mention of Mr. Moore in its publications or website other than an occasional joke about his some of his experimental keyboards. I felt that FIG should be open to providing publicity for the fact that Mr. Moore had written his own CAD systems in Forth, developed new generations of his language about every five years, and designed chips that were faster than Intel's top of the line chips at that time. But that is another story, behind the FIG door, that will get its own web page in the future.

At the same time I was working with Charles Moore, Dr. Carter, and a number of other people at the iTV Corporation which was developing low-cost Internet appliances, browsers, email, and other application software for devices based on the i21, a licensed variation of my F21 Forth chip. Chuck worked on chip design, I worked on hardware testing and became programming manager for a team of a dozen Forth programmers using ANS Forth and the chip's native machineForth.

Skip Carter worked as a contractor and was responsible for writing our network protocols. The company's managers wanted some of our application written in portable ANS Forth, and like much of the Forth Scientific Library the protocol code was based on a port of available C code. We had weekly programming staff meetings to discuss changes requested by management and implementation details. These often became heated discussions as there developed two clear schools of thought. One was represented by the President of FIG, Dr. Carter, and the other by the Vice President of FIG, myself.

The ANS Forth programmers were concerned about portability and were not as concerned about efficiency. The machineForth programmers were writing for our hardware and did not care about writing code for anything other than the product that we had to develop to keep the company going. This resulted in much conflict regarding what was considered good programming style. While programmers in each camp delivered working code, and the overall result met the product objects and was well received by management and our test users the debates about what was good style in Forth went on for years. Similar debates continue today, and I hope will continue in the future, although without the shouting or name calling that we sometimes see.

Management made it clear that the product was being marketed as being lower cost and simpler than the competitors products. It was an embedded application in an Internet Appliance that we were building and we did not want the software to become too bloated or inefficient. Forth has two stacks in order to support a different form of factoring of code than C. Forth words, are similar to C functions, but are generally much shorter using from one to a few lines of code for the definition of a word. I was somewhat concerned about our network code because much of the code in the Forth Scientific Library was a translation of C code into Forth with little refactoring into Forth style. As our network code evolved one of the things we often heard was that, "It is not a coincidence that this Forth code has exactly the same factoring and naming of functions as the C code."

As mentioned earlier C-style programming is very much dependent on C-style data structures. In order to facilitate porting C code into the Forth Scientific Library, and into our network code, one of the important files in the FSL was C-style structures. There is an online document explaining A rationale for Structures in the Forth Scientific Library.

The library file which got pasted into our embedded application with little change was structs.fth (Yes, 27K of actual Forth code for your review). One of the changes that was made was to the word **?member-error** from

```
: ?member-error ( m-id s-id -- )  \ raise an error if s-id and m-id
```

```
                                      \ do not match
         OVER OVER

         <> IF  SWAP  ." Wrong member of structure, STRUCT = " U. CR
                      ." , Member = " U. CR
                  ABORT
            THEN

        2DROP
;
```

which relies on an **ABORT** which was not appropriate for our embedded application to

```
: ?member-error ( m-id s-id -- )
 2DROP
;
```

When I first looked at this example of ANS Forth code I was more than a little concerned. I sped up the code by a factor of about five in about ten minutes. I then spent another hour on the code and sped it by another factor of about one hundred by writing a few simplified key routines in machineForth instead of ANS Forth. Since we were only supporting a dial-up modem at the time the 500 times speedup for the structures code was more than sufficient for our needs. I made some notes and used it as an example in our next staff meeting of what I had been talking about for some time.

```
: resolve-structure-member ( s-id s-addr m-base -- m-id m-addr )
         ROT >R
         DUP 2@ SWAP R>
         ?member-error
         SWAP [ 2 CELLS ] LITERAL + @
         ROT ROT +
;
```

I showed the actual timing of the routine as compiled in our application. I pointed out that there was a small difference between this and what we would get if **2DROP** was substituted in this word for **?member-error** rather than what we had which was a new definition of **?member-error** as **2DROP** because all that would remove from the definition was the overhead of one thread and unthread. However what I felt was more important was that there was a lot of stack juggling in this word in order to pass the two arguments to **?member-error** which then simply discarded. Why did it carry these arguments just to drop them?

I pointed out a number of words that had **ROT**s and **>R**s and **R>**s and **DUP**s and **SWAP**s that simply were not needed. They were there to carry the weight of arguments that were not being used, and were just going to be discarded later. I pointed out that if one simply removed the **?member-error** which was now just **2DROP** that one could then remove many of these words to juggle dummy arguments and that the overall result was that the code was about five times faster with exactly the same functionality. I pointed out that this sort of simple cleanup of the library ANS Forth was something that I would expect from a beginning ANS Forth programmer, let alone from the President of FIG. I pointed out that this was code that was used all over the place in the network code and that it was important that it not be too inefficient.

I went on to show that with a little more effort, about one hour, that I was able to take the greatly simplified ANS Forth code and convert many of these primitive functions to machineForth for a much larger speedup as well as further reduction in the size of this module while keeping the exact same functionality in our product.

I said that I wished that our programmers would pay more attention to our goals of writing a small embedded application rather than relying on the machineForth programmers to clean up their code.

But we had these kind of discussions week after week for years. We always seemed to come back to the issue that the ANS Forth Standard had been designed to facilitate the use of portable libraries like this, and that they were considered good style by many ANS Forth programmers and being promoted by FIG and commerical Forth vendors. The machineForth programmers did not consider it good Forth style. You are free to form your own opinion.

One of our programmers worked in Russia and contributed some excellent machineForth routines to the project. He had neither a copy of our expensive Windows development system, nor an actual Forth chip Internet board. So he modified an F21 emulator to emulate the entire board with modem, and flash memory so he could debug code. The first time that he looked at some of the ANS Forth code that came from the FSL he asked in all seriousness, "Is this C code or Forth code?" He had never seen Forth code full of .h files, neither had I until then.

I mentioned this example, and many other similar examples that we had at iTV in the usenet group comp.lang.forth a few years ago. The chairperson of the ANS Forth Technical Committee, who had nominated Dr. Carter for FIG President, and voted for him and repeatedly praised his Forth Scientific Library project, even including it in Forth Inc.'s commercial releases said that she did not know who this ANS Forth programmer was.

When I explained things such as my concern that the portable ANS Forth application libraries that she had praised so often were not in my opinion examples of what makes Forth shine and that the problem with them was that they were simply too much like C and being used like C my argument were characterized as irrational and intentionally untrue. My concern was, and still is, that many things were introduced into ANS Forth to facilitate and promote this sort of pasting of less than optimal code from libraries rather than the kind of Forth that I characterize as Thoughtful Programming. Ms. Rather wrote in comp.lang.forth on 12/13/2000:
The stated purpose of ANS Forth is (Section 1.1):

"The purpose of this Standard is to promote the portability of Forth programs for use on a wide variety of computing systems, to facilitate the communication of programs, programming techniques, and ideas among Forth programmers, and to serve as a basis for the future evolution of the Forth language."

As one who attended every meeting and participated in exhaustive discussions of the mission of the TC, I believe this is a fair and accurate statement of our objective. I see nothing in there about appealing to C bigots. So, you're welcome to say "some people perceive that ANS Forth..." but please do not make false and misleading statements.

I did not see where I had made any false statements. I also considered it misleading to characterize my arguments as saying that the purpose of ANS Forth was to appeal to C bigots. I had not called anyone names or used this phrase. Hopefully this essay will help clarify that my concerns about Forth style and the future evolution of the Forth language.

If nothing else, one may note that the structs.fth program in the FSL was changed and improved after my raising the issue that the original version had room for improvement. I also thought it funny that I got at least one email explaining that this person just didn't know what I had been talking about in all my comments that some people were trying to add more C-style stuff to Forth until he realized that he had made a contribution labeled C-Style something-or-another in ANS Forth.

Jeff Fox
10/27/02
Related references:

ANSI Forth is ANTI Forth Jeff Fox 2/28/99, **Chuck Moore** 7/26/98, 3/5/99
Thoughtful Programming Long Essay on Forth by Jeff Fox
Forth Methodology Jeff Fox 8/2000
Essential Forth Jeff Fox 7/16/02
Levels of code in Forth programming Jeff Fox 5/29/02
www.colorForth.com Charles Moore's website
ANSI Forth X3J14
Forth Interest Group