Bilingual Programming System DSSP+Forth

Eugenia N.Lyakina, Dr. Sergei A.Sidorov
NIISI RAS, Russia

For about 20 years the programming systems Forth and DSSP (Dialogue System of Structured Programming) existed and developed in parallel ways. Today they look alike, but inside there are some basic differences. What is the depth of these differences? Are there fundamental and hardly surmountable misfits in these systems? The quantitative answer was obtained while creating Bilingual Programming System, which is capable to understand both languages - Forth and DSSP. We need such a system as the base for the development of the firmware conforming to IEEE Std 1275-1994 «Open Firmware».

This paper tells about the reasons of creating Bilingual Programming System. We analyze concrete differences in the languages and internal mechanisms, describe way of the system development and give some estimations of coincident and different parts.

Introduction

DSSP was created as a side branch of Forth, which adopted its best ideas. Then DSSP developed in its own way. At one of the previous conferences we presented our paper with comparative analysis of both systems [1]. In spite of long independent development both systems are close enough. Each of them has merits and demerits, but we shall not discuss it here. This paper is devoted to the examination of the existing differences and to the search of way to merge DSSP and Forth.

One of DSSP application sphere is creating firmware for newly workable universal and embedded computers. We have accumulated wide experience and big volume of programs. Now there are some traditions and use-proven solutions in this application. Our firmware system DPROM [2] has been successfully used for more then 10 years. Forth has also been used for a long time for these purposes. In 1994 Standard IEEE Std 1275-1994 «Open Firmware» [3] was published. It is based on Forth system with the Forth language as part and parcel. «Open Firmware» is the real way to easy use wide interface cards collection which follow Standard. We wanted to build firmware system with «Open Firmware» opportunities and to save DPROM facilities. The way was to modify and improve DPROM firmware to conform IEEE Standard.

Some words about fundamental parts of «Open Firmware»:

1. Device tree - the main structure which contains information about computer configuration. The node of the tree corresponds to the device and keeps the device properties, methods (procedures) and parameters. Device tree is built (or, more exactly, is completed) during computer initialization as the result of probing, that is the search of plugged devices.
2. Device interface, that is the instrument for FCode reading, transforming to threaded code and executing procedures obtained.

3. Client interface, that is the support of application calls. Usually this application is Operating System, which makes calls to get information from the device tree about actual computer configuration.
4. User interface, which supports interactive mode. By this interface user can execute firmware commands including Forth, debugger, parameters editing and so on.

After careful studying of Standard it became clear that all its conceptual and algorithmic parts could be easily implemented in DSSP. Forth is needed for FCode supporting and as the user interface. We decided to extend DSSP abilities in such a way, that DSSP could execute Forth programs. Now we consider our method and results.

1. Structure and Components of Programming Systems

It is evident that DSSP and Forth are relatives. For our purposes it has meaning to determine not functional, but structural components of the systems, and to analyze them one by one. We determine the following components:

- Language
- Threaded code
- Dictionary
- Outer interpreter and compiler
- Data

Let's compare these components in detail and inspect differences.

2. Language

As the Forth language we considered commands described in Standard ANSI X3.215-1994 «Programming Languages - Forth» [4] in chapters «Core words», «Core extension words» and additional words, listed in «Open Firmware» Std, excluding ones related to firmware itself. We counted 180 such words. After comparison of word lists we defined 5 command groups. In some cases this division is very relative.

Category	Number	Examples
The same by name and by function	16	+ - 1+ > = max
The same by function but with different names	36	copy drop over and negate depth
Resembling by function or easy to implement	25	move find tuck
Different by function but the same by name	12	! ‘ value
Forth words absent in DSSP	91	if loop exit 2* roll

The first category is obvious because it consists mostly of mathematical signs and standard functions. Unfortunately it is the smallest group.

The second list also natural due to the same structure of stacks, memory and common language principles. DSSP follows the rule to give short names for reducing

manual work. For example, copy and drop in DSSP are named C and D, negate - NEG, over - C2 (Copy 2nd) and so on. Not only primitives are included to this list, but some more complex words, such as «key» - «TRB», «word» - «WRD», «[']» - «‘‘».

Resembling or easy to implement words in most cases are different in parameters order or number. Here are some examples of such Forth words and their implementations in DSSP (comments in examples are given in Forth notation to avoid ambiguity; where possible DSSP words are displaced by Forth words):

Forth	DSSP	Implementation
move	!SB	(addr1 addr2 n --) : move swap !SB ;
tuck		(x1 x2 -- x2 x1 x2) : tuck swap over ;
parse	CIWD	(char -- addr len) : parse !DELIM WRD ACURR WLEN ;

The most unpleasant is the word group of the same name with different action. So, command «!» in Forth means «store subtop at address from top», but in DSSP it means «store top to variable which name follows the !»; DSSP word VALUE defines named constant unlike Forth, where it defines a kind of variable. This force us to separate such words into an additional dictionary.

The lagest category consists of words absent in DSSP. Many of them ought to be implemented as primitives. Others are implemented by «long» definitions - 5-6 or more words. Above all there are the control flow words originally different in Forth and DSSP (22). Next - double precision arithmetic and double-word manipulations in stack (16). Then, some words related to outer interpreter and compiler (21). At last, scattered commands for stack manipulations, arithmetic, input/output, number and string transformations (32).

Thus, DSSP language requires numerous appending because about half of Forth words are absent in DSSP. At the same time implementation of Forth words is not difficult and consumes a little space. Common space evaluation will be given at the end.

3. Threaded code

The Forth Standard does not direct to use any appointed threaded code. The DSSP version we work with is built as virtual machine (C-written) with linear virtual address space. In this model two kinds of threaded code are implemented. The first, direct threaded code is simple direct address references in virtual space (it can be named relative in actual computer space). The second is based on byte-coding like FCode with intermediate code-address translation table. According to our estimation this approach allows to spare some amount of memory in large applications (several hundreds of definitions). Firmware does not require speed, but compactness.

All actions concerning internal structure of threaded code use special procedures which hide details. Thus we can choose one kind of threaded code depending on external conditions.

4. Dictionary

Dictionaries in Forth and DSSP serve the same purpose, but they are made differently. DSSP dictionary [6] is the sequence of named sub-dictionaries, each of them consists of dictionary entries. A dictionary entry keeps the word header: name and flags, and the pointer to procedure body. Procedure FIND searches word from the dictionary end, checking sub-dictionaries one by one, except those marked as shut. If the word is undefined still, it is followed by the table of addresses where this word is used. The table supports top-down programming technique. DSSP dictionary does not contain any internal references and its location in memory is defined by pair of pointers - the beginning and the end. This provides two important features: dictionary can be easily moved in memory and temporary entries can be deleted from dictionary to avoid conflicts and reduce memory consumption.

Forth dictionary consists of sub-dictionaries too but the procedure FIND chooses search order in another way. Necessary sub-dictionaries are arranged to Context list and this list is used by FIND. Dictionary entries are linked as a list. This feature allows to grow any sub-dictionary at any time. List-based dictionary supports various manipulations with the dictionary, but it is difficult to move or clean it.

From the most programmers point of view either DSSP or Forth dictionaries are good. In fact we usually use dictionary manipulations only when assembling large programs to avoid name collisions or to form any sub-language for interactive system. In some cases, however, dictionary internal structure is used in essence. In particular «Open Firmware» device tree is based on Forth dictionary lists. The best way in this situation was to choose Forth dictionary as a starting point and add mechanism for top-down programming. It was easy: if T-flag in word header is set, this word is followed by the head of list of addresses where this word is referenced. The list itself contains address and the pointer to the next element.

Thus, we saved one of the main DSSP features - top-down programming in new system and obtained all Forth dictionary functionality.

We separated words with different functions but the same names into different sub-dictionaries. In Forth mode system uses one sub-dictionary, in DSSP mode - another.

5. Outer interpreter and compiler

These two components are the «face» of interactive systems. We found following distinctions of outer interpreter and compiler in DSSP and Forth.

Comments processing: Forth inputs input stream without filtering and the comments are processed by word «(». In DSSP comments are sifted out while reading next string. Interpreter (and compiler) gets «plain» text.

Forth comments	DSSP comments
(x y -- z) : a ... ;	: A [x,y] ... [z] ;

This problem we solve by analyzing the mode flag: if DSSP mode then comments filter is ON, otherwise OFF.

While compilation Forth uses *immediate* words, i.e. active elements which do some additional work during compilation process. These words implement control commands, text processing and some other. This principle allows to simplify the compiler itself and distribute special jobs among agents. DSSP keeps to the rule «One word of program text corresponds to one word of code» and no additional actions are needed. Special processing is required only for texts; compiler has a branch for these cases.

It is hard to reduce such a considerable difference in one program. We implemented separate outer interpreter and compiler for Forth. They use already existent DSSP procedures for input stream reading and text parsing, dictionary search and entries creation, building of procedure body, memory allocation and so on. In fact, only top level must be programmed, all infrastructure is ready.

DSSP supports *top-down* programming unlike Forth.

We extend Forth dictionary with top-down supporting mechanism and now this problem is solved. Moreover, some modifications of Forth compiler can allow to apply top-down technique in Forth programs.

6. Data

DSSP data system is based on prefix access methods [5]. Simple Forth variables are mapped into named number constants with address as its value. Forth «value» and «defer» with store command «to» are mapped into usual DSSP variables:

Forth	**Implementation in DSSP** (Forth-style comments)
variable A	: variable 4 GET_MEM (addr) CONSTANT ;
(x) value B	: value VAR MEMPTR 4- (x addr) !TL ;
defer N	: defer ‘‘ UNINIT (proc.addr) ACT value ;

Results

Now we have a hybrid system DSSP+Forth with temporary name BPS - Bilingual Programming System. It can work in two modes - either DSSP or Forth. DSSP program can use many words from Forth collection, and vice versa. Mode switch words «FORTH» and «DSSP» do simple work: set or reset mode flag and connect one of two alternative sub-dictionaries with words of the same name.

There are some changes in DSSP language concerning dictionary. Now we use Forth words for these purposes. It is not hard to imitate DSSP dictionary command set, maybe with some restrictions, but it is not necessary today.

We implemented Forth words from ANSI Standard parts «Core», «Core extension» and words listed in IEEE Std 1275. It is enough to conform «Open Firmware».

What is the cost of this result? Programming itself was very easy, most time was devoted to understanding internal Forth mechanisms, such as immediate words,

outer interpreter details, «CREATE ... DOES>». Code increase is about 35-40% relatively to original DSSP, depending on target processor and compiler. Some figures:

CPU	Original DSSP length	DSSP+Forth length	increase
MIPS R3000	77 KB	107 KB	39%
Sparc	69 KB	94 KB	35%
MC68K	65 KB	91 KB	40%

We consider these figures satisfactory enough. Comparatively with full firmware system which size we evaluate as about 500 KB, these additional 25-30 KB are the low cost for convenience.

References

1. Sidorov S.A., Shumakov M.N. DSSP and Forth. Compare analysis. // 12th EuroFORTH conference on the FORTH programming language and FORTH processors. St.Peterburg, Russia. 1996.
2. Ljakina E.N., Sidorov S.A., Shumakov M.N. Applications of DSSP. // 12th EuroFORTH conference on the FORTH programming language and FORTH processors. St.Peterburg, Russia. 1996.
3. IEEE Std 1275-1994 - IEEE Standard for Boot (Initialization Configuration) Firmware: Core Requirements and Practices.
4. ANSI X3.215-1994 - Programming Languages - Forth.
5. Sidorov S.A. Data in DSSP - prefix access in postfix language. // EuroFORTH'97. Conference proceedings.- Oxford, England, 1997.
6. Sidorov S.A. Top-down Thinking and Top-down Writing in DSSP. // EuroFORTH'98: 14th Euroforth Conference, Schloss Dagstuhl, Germany, 1998.