Multiple Language Programs Made Easy

Howerd Oakford, Inventio Software

12 Aug 98

Abstract :

A simple extension to the Forth compiler allows different language (English/Spanish/German) versions of the same program to be produced from identical source code. Word processor compatible translation files are used to allow translation of text messages by non-programmers.

Forth is alive and well in several niche markets. Getting the maximum performance from your hardware, providing extensible user interfaces (scripting languages) and the creation of customised compilers are three areas which spring to mind.
I was asked recently to provide a Spanish version of a program that I had previously written using 8051 chipForth, and rather than mechanically copy and edit the source I decided to try to automate the process as much as possible, and make it generally applicable to any Forth program. It was the customised compiler facet of Forth that allowed me to do this in a very simple way.

The application in question is an embedded 8031 with 32K battery-backed RAM, 32K EPROM, keypad, LCD, which together made up the electronics of a Moisture Meter to measure the percentage moisture content of grains such as wheat, barley and oats. This product is in fact the direct descendent of the unit that I designed circa 1979 - in those days the processor board used an 1802 with 2K of EPROM and 32 bytes of RAM. The LCD today has also gone from 4 digits to 2x20 characters, and the keypad from 8 to 16 keys. In step with the improvements to the hardware, the software has changed too : the original was programmed in 1802 microForth using an 1802 based development system - now its 8051 chipForth hosted on a PC. The application software used to just measure moisture and weight, with simple averaging, now by using an Acoustic Coupler Modem (another 8051 chipForth product!) calibration curves can be transferred to and from units by phone using a PC based program. The unit can also be operated and calibrated remotely over the phone, and it is even possible to run chipForth remotely should this ever become necessary!

The source for the program occupies roughly 120 blocks, and to provide an EPROM with the LCD and RS232 messages translated into Spanish (or any other language) the following additional code is added to the chipForth system :

First, some database and general support words are added to the Host's Forth system. This part of the program accesses a file called SPANISH.TRN using block buffers. A file based system would be similar. All that is required is a linear contiguous list of 128 byte records, indexed by number. The file is created externally using DOS and is of fixed length (say 32K bytes).

Second, a modified version of the target compiler's **STRING** is created, called **+STRING** .
+STRING performs the same actions as **STRING** , except that after the string is fetched using **WORD** the vector **?LANG** is executed to perform the translation. Three actions are defined for **?LANG** :
MAKEFILE copies the English text into two consecutive half-records (of 64 bytes each),
ENGLISH sets up the default no-op to use the English text straight from the source,
SPANISH sets up the compiler to compile the translated string from the file, rather than the original source text.

Thirdly, the Host's cross compiler versions of **dot"** and **"** are modified to use **+STRING** instead of **STRING** .

To use the program, load the cross compiler by typing **COMPILER LOAD** , then type :

MAKEFILE prog LOAD \ prog is the target program's load block

This copies every string compiled by **dot"** or **"** into a record in the file SPANISH.TRN, duplicated twice into each 128 byte record.

Give the file SPANISH.TRN to the person who will translate it. They can use any word processor capable of handling conventional DOS text files, as a carriage return and linefeed is included for each text entry .
Each text item appears on two consecutive lines - the human translator simply edits the second line to its translated form. The translated text must occupy the same space on the LCD as the original.
The edited file is then returned to the chipForth system.

ENGLISH prog LOAD \ creates the English version, straight from the program source

SPANISH prog LOAD \ creates the Spanish version, using the SPANISH.TRN file.

The entire translation program occupies 5 blocks (plus 5 more for the shadows) , and can be printed out on two sheets of A4 paper.
The source code following is for 8051 chipForth, but the principles are easily adapted to any Forth system. (Note : **[ASCII] X** compiles hex 58 as a literal , **I** is the same as **R**@).

Extensions to the system could include :

- the record structure could be modified for many languages in one file,
- all language variants could be compiled - which one to display is selected at run time,
- the simple indexed database could be replaced by one which searches the file for the required text. There could be a prompt displayed to request a translation if it could not be found in the file. This would add some complexity to the program, however.

The system illustrates several principles :

- Having access to the source code for a well written and highly factored product such as chipForth makes customised compiler actions very easy to implement.

- If the internal structure of a program is kept simple, it is easy to add interfaces to external systems which are much more complex. The program uses blocks which are tweaked to be readable by a word processor.

- If a file is of fixed length, the difference between a file and block based system is just a carriage return and linefeed at the end of each line. If records are of fixed length and number, blocks allow a one line database to be written (see **>FILE**). Blocks can be read by a word processor if crlf is added for each line.

- It is possible to create any number of language variants of any Forth program using identical source, with a translation file for each language or set of languages.

- The constraints of the system such as fixed field sizes on the LCD and fixed number of records allow vast simplification compared to a generic solution

- The boundaries between the source code, compiler and editor are becoming less clearly defined.

Howerd Oakford 12 Aug 98
Inventio Software - Specialist Instrumentation Design and Programming
Phone +44 1344 457057 Fax +44 1344 640546 Email inventiosoftware@compuserve.com
WWW http://ourworld.compuserve.com/homepages/inventiosoftware

```
69              Unit 0  Part 0  Abs    69
 0 ( Foreign language translation )
 1 VARIABLE ?LANG
 2
 3 : ENGLISH   0 ?LANG ! ;  ENGLISH
 4
 5 VARIABLE TO:
 6
 7 : $COPY ( source dest)   TO: !  COUNT  40 MIN  >R
 8   TO: @ 1+  I  MOVE  R> TO: @ C! ;
 9
10 VARIABLE NUMM  0 NUMM !
11
12 : >FILE ( n - a) 8 /MOD 2100 + BLOCK SWAP 128 * + ;
13
14
15
```

```
70              Unit 0  Part 0  Abs    70
 0   ( Foreign language translation )
 1 : TRANSLATE   NUMM TALLY
 2   ( CR ." >>> "  DUP COUNT TYPE )
 3   NUMM @ >FILE 2+ 64 +
 4   ( CR ." *** "  DUP 20 TYPE )
 5   OVER COUNT  MOVE
 6   ( CR ."     "  DUP COUNT TYPE )
 7   ;
 8
 9 : SPANISH   7 UNMAP  7 MAPS" SPANISH.TRN"  2100
10   MODIFY  ['] TRANSLATE ?LANG !  0 NUMM ! ;
11
12
13
14
15
```

```
71              Unit 0  Part 0  Abs    71
 0   ( Foreign language translation )
 1 ( EXIT  : This overwrites the file SPANISH.TRN !!! ( *** )
 2 : (MAKEFILE)   CR ." >>> "  DUP COUNT TYPE
 3    NUMM TALLY  DUP COUNT  NUMM @ >FILE >R
 4   ( mark line) I 64 [ASCII] . FILL
 5   ( CRLF)  13 I C!  10 I 1+ C!
 6   ( text )  I 2 +  SWAP  MOVE
 7   ( Default translation) I I 64 + 64 MOVE
 8   R> DROP  UPDATE ;
 9
10 : MAKEFILE   7 UNMAP  7 MAPS" SPANISH.TRN"
11  2100 MODIFY  ['] (MAKEFILE) ?LANG !  0 NUMM ! ;
12
13
14
15
```

```
369             Unit 0  Part 0  Abs   369
 0
 1 ?LANG  contains the vector to select the required
 2  language text
 3 The default is ENGLISH which sets a null action
 4
 5 TO: is a "local" variable, used for readability
 6
 7 $COPY  copies a counted string from source to
 8  destination, up to a maximum of 40 characters
 9
10 NUMM contains the current count of string number
11
12 >FILE  returns the address of text string n in a block
13   buffer
14
15
```

```
370             Unit 0  Part 0  Abs   370
 0
 1 TRANSLATE  is the vectored action which inserts the
 2   translated text into the target program
 3
 4 SPANISH  maps the Spanish translation file into block
 5   2100 and above and sets the translation action for
 6   +STRING
 7
 8  The text in parentheses is test code to display the
 9  strings being compiled into the file
10
11Note that NUMM is pre-incremented so that the top
12  two lines can be used for a title.
13
14
15
```

```
371             Unit 0  Part 0  Abs   371
 0
 1 (MAKEFILE)  is the vectored action which creates a
 2   default translation file, with each text entry repeated
 3   twice.
 4   Text which is not manually modified in the file will then
 5   be the original English test.
 6
 7 MAKEFILE  sets up the Spanish translation file and the
 8  file creation action for STRING
 9
10 Note this block is only loaded once to create the initial
11 file.
12 The paranthesis in front of the EXIT in line 1 is then
13 removed to prevent accidental loading.
14 Line 7 may be commented out so that only the English
15 lines are overwriten.
```

```
250             Unit 0  Part 0  Abs   250
 0   ( Target assembler)
 1 : STRING ( c)   WORD  DUP C@ 1+  DUP GAP
 2    HERE OVER - SWAP CMOVE ;
 3
 4 : +STRING ( c)   WORD  ?LANG @EXECUTE_ DUP
 5    C@ 1+  DUP GAP HERE OVER - SWAP CMOVE ;
 6
 7 : EQU ( n)   CONSTANT ;
 8 : CODE   (CREATE) -3 GAP  ASSEMBLER ;
 9 : LABEL   HERE LOG  EQU  ASSEMBLER ;
10
11 ASSEMBLER DEFINITIONS   HEX 1F1F WIDTH !
12   DECIMAL 231 238 THRU
13 HOST DEFINITIONS   ( HEX 0303 WIDTH ! )
14
15 : USE ( a)   ><  EHERE 2- E@ 1+ T! ;
```

```
550             Unit 0  Part 0  Abs   550
 0 CODE  creates a target CODE entry
 1 EQU  (for equate) defines a host constant ( CONSTANT
 2  will be redefined)
 3 LABEL  names the current position in the target
 4   dictionary
 5 STRING  constructs a string in the target dictionary
 6
 7 +STRING  constructs a string in the target dictionary,
 8  but has an additional action which can be used to
 9  translate the string
10 BEGIN  and U) as well as all other assembler words are
11  redefined  to compile into the target dictionary
12 USE  is also so redefined
13
14
15
```

```
209             Unit 0  Part 0  Abs   209
 0   ( Ascii display formatting)
 1 : ?R@ ( t - | a)   R> SWAP  R> DUP COUNT  + >R
 2      SWAP IF  SWAP >R EXIT  THEN  2DROP ;
 3
 4 FORTH : [ASCII]   32 WORD 1+ C@  [COMPILE]
 5    LITERAL ;   TARGET
 6
 7 : dot"  1 ?R@ COUNT >TYPE ;
 8 FORTH : ."   COMPILE dot"  [ASCII] " +STRING ;
 9   TARGET
10 : string ( - a)   1 ?R@ ;
11 FORTH : "   COMPILE string  [ASCII] " +STRING ;
12   TARGET
13
14 HOST
14
```

```
509             Unit 0  Part 0  Abs   509
 0 String output words include:
 1 ?R@  tests a truth value, if set it returns with an in-line
 2    string address; otherwise it exits from the calling word
 3    (usually .")
 4
 5 [ASCII] compiles the ASCII value of the following
 6  character.
 7 dot"  is the run-time code for ."  It types the message.
 8 ." is a host definitions which references the just-defined
 9    run-time code.
10
11 string  pushes the address of and steps past a string.
12 " compiles a string literal into a definition.
13
14
15
```

The first few lines of file SPANISH.TRN :

```
( GrainPro V1.0 26 Jun 98 English/Spanish translation)
( English is first, folowed by Spanish on alternate lines )
                    ........................................
                    ........................................
SAMPLE TESTS        ........................................
TEST DE MUESTRA     ........................................
PRINTER MENU        ........................................
MENU DE IMPRESORA   ........................................
BULK DENSITY MENU   ........................................
MENU PESO ESPECIFICO........................................
CALIBRATION MENU    ........................................
MENU CALIBRACION    ........................................
etc etc....
```

The dots mark out the number of characters required for each string. This is constrained by the position of the text on the LCD - the translated version must occupy the same size field.