Forth-83

TESTING TOOLKIT

PHIL KOOPMAN, JR. - WEXFORD, PENNSYLVANIA

One of Forth's strong points is its support of interactive development and testing. Sometimes, however, interactive testing is not enough. During the development of low-level software for the RTX family, we wanted a method to create a permanent record of test cases for Forth words. This record serves as documentation for users and maintainers. In addition, a full suite of test cases for a program provides a way to be sure that a change in one part of the program does not disturb other parts of the program.

How to Use It

Each test case consists of code that places elements on the data and return stacks, creates and executes a test definition, then verifies that the correct results were placed on both stacks. For example, a test case for the word DUP would be:

The test case can be any sequence of Forth words.

```
DS( 1111 --
RS( --
TEST: DUP ;DONE
-- )RS
-- 1111 1111 )DS
```

The first line of the test case specifies that the data stack input to the test is the number 1111. The second line specifies that no elements are to be placed onto the return stack. The third line creates and executes a temporary Forth word with a body of DUP, carefully handling the data and return stack contents before and after the test. The fourth line specifies that no values should

```
\ Forth testing support
\ By Philip Koopman Jr., for Harris Semiconductor
\   Derived from test code used for the RTX chip family
\   Developed on F-TZ (an F-PC and F-83 derivative) version 3.X11

VARIABLE #STACK -1 #STACK !   \ Saves number of stack elements for testing
CREATE R-SAVE 8 ALLOT         \ Note: F-TZ uses 32-bit return addresses!

: GET-DEPTH  ( ..stack.stuff.. - ..stack.stuff.. )
  DEPTH  #STACK @ --  #STACK ! ;

: DS(    ( -- $BAD1 $BAD2 )
  \ Init RS to -1 so that '-' will know it is a DS input
  \ Uses hex 0BAD1 and hex 0BAD2 as sentinel values for DS
  -1 #STACK !     $BAD1 $BAD2 ;

: RS(    ( -- $BAD3 $BAD4 )
  \ Uses hex 0BAD3 and hex 0BAD4 as sentinel values for RS
  DEPTH #STACK !  $BAD3 $BAD4 ;

: --     ( n1 n2 n3 .. n.n - n1 n2 n3 .. n.n sentinel )
  #STACK @  0<  NOT    IF ( if RS( )  GET-DEPTH   THEN ;

: ?DATA    ( n1 n2 -- )
  = NOT  ABORT" DATA STACK ERROR" ;

: ?RETURN  ( n1 n2 -- )
  = NOT  ABORT" RETURN STACK ERROR" ;

: --     ( --)
  DEPTH  #STACK ! ;

: PERCOLATE  ( r1 n.n .. n1 --  n.n .. n1 r1 )
  #STACK @  ROLL  -1 #STACK +! ;

: )RS        ( r.n .. r3 r2 r.1  n1 n2 n3 .. n.n -- )
  GET-DEPTH   #STACK @
  IF  BEGIN  PERCOLATE  ?RETURN   #STACK @ 0= UNTIL  THEN
  $BAD4 ?RETURN  $BAD3 ?RETURN  -1 #STACK ! ;

: )DS        ( r.n .. r3 r2 r.1  n1 n2 n3 .. n.n -- )
  GET-DEPTH   #STACK @
  IF  BEGIN  PERCOLATE  ?DATA     #STACK @ 0= UNTIL  THEN
  $BAD2 ?DATA    $BAD1 ?DATA    -1 #STACK ! ;

: REVERSE    ( n.1 n.2 .. n.n  n  --  n.n .. n.2 n.1 )
  DUP 0> IF  0 DO   I ROLL   LOOP  ELSE DROP THEN ;

: INIT-TEST  ( ..DS.stuff..  ..RS.stuff..  -- ..DS.stuff.. )
             ( RS:   -- ..RS.stuff.. )
```

be left on the return stack, and generates an error message if this is not the case. The fifth line specifies that two values of the number 1111 should be returned from the test, again generating an error message if this is not the case. It is very important that the test cases be written in exactly this order, with no missing items, for proper operation.

The body of the test case between `TEST:` and `;DONE` can be any sequence of Forth words, including primitives that manipulate the return stack. The words `INIT-TEST` and `FINISH-TEST` are automatically compiled with the test case to handle the data and return stacks for proper execution.

In order to be sure that a word is working properly, it is not enough to simply place the required number of parameters on the stack and then see if the correct results are returned. The problem is that a word may cause unexpected side effects (such as corruption of elements on the data and return stacks) that are not detected immediately. In order to handle this case, the test words place two "sentinel values" onto both the data stack and the return stack, then check to ensure that no corruption has occurred. While side effects are usually not a problem in high-level code, they can easily create problems when dealing with assembly language or microcode word implementations.

Ideas for Further Refinements

The test capability presented here is rather simple, in order to keep the code (somewhat) understandable. Features that could be added to improve its usability include: allowing `RS ( ) RS` to be optional, so tests that deal only with data stack operations could automatically generate and test return stack sentinel values; more sophisticated error messages that show exactly what is wrong with a stack when an error does occur; methods to ensure that only desired memory locations are modified for words that perform fetches and stores; and methods to ensure that only desired on-chip registers are modified for assembly language definitions.

The code is written for F-TZ, a version of F-PC, developed by Tom Zimmer. F-PC is a descendent of F-83, but allows using a dictionary space of greater than 64K bytes. The code presented should be relatively

(Continued on page 41.)

```
  CR ." TEST-"
  #STACK @  0<  ABORT" You must specify both DS( and RS(."
  R> R-SAVE !  R> R-SAVE 2+ !          \ Save return address
  #STACK @  REVERSE
  BEGIN  #STACK @  0> WHILE  >R  -1 #STACK +!  REPEAT
  R-SAVE 2+ @ >R  R-SAVE @  >R   ;     \ Restore return address

: FINISH-TEST  ( ..DS.stuff.. -- ..DS.stuff.. ..reversed.RS.stuff.. )
               ( RS:  ..RS.stuff.. -- )
  R> R-SAVE ! R> R-SAVE 2+ !           \ Save return address
  \ Transfer return stack contents onto data stack for later compare
  0 >R
  BEGIN  R> R>  SWAP 1+ >R   DUP $BAD3 = UNTIL
  R> REVERSE
  R-SAVE 2+ @ >R  R-SAVE @  >R         \ Restore return address

  ." -DONE"  -1 #STACK !  ;

\ TEST and DONE use F-TZ specific words to compile a short
\  definition containing the word to be tested, execute that
\  definition, then FORGET it from the dictionary.
\ This borrows a compilation idea from Rick van Norman's RTX test code
CREATE MARKER  4 ALLOT
: TESTER ;
: TEST:   ( --)
   XHERE  2DUP MARKER 2!  PARAGRAPH +  DUP XDPSEG !  0 XDP !
   XSEG @ --  ['] TESTER >BODY  !
   COMPILE INIT-TEST  ] ;


: ;DONE
   COMPILE FINISH-TEST  COMPILE EXIT
   STATE OFF  TESTER  MARKER 2@  XDP ! XDPSEG !  ;
IMMEDIATE

\ Test ROT for proper operation
  DS( 1111 2222 3333 --
  RS( --
  TEST: ROT ;DONE
  -- )RS
  -- 2222 3333 1111 )DS


\ Test >R for proper operation
  DS( 5555 --
  RS( --
  TEST:  >R  ;DONE
  -- 5555 )RS
  -- )DS


\ Any combination may go between TEST: and ;DONE
  DS( 1111 2222 3333 --
  RS( 7777 2222 9999 --
  TEST:  SWAP R> ROT >R  ;DONE
  -- 7777 2222 3333 )RS
  -- 1111 2222 9999 )DS


\ Null test to be sure it works
  DS( --
  RS( --
  TEST:  ;DONE
  -- )RS
  -- )DS
```

International Forth BBS's

- Melbourne FIG Chapter
  (03) 809-1787 in Australia
  61-3-809-1787 international
  SysOp: Lance Collins
- Forth BBS JEDI
  Paris, France
  33 36 43 15 15
  7 data bits, 1 stop, even parity
- Max BBS *(ForthNet link*)*
  United Kingdom
  0905 754157
  SysOp: Jon Brooks
- Sky Port *(ForthNet link*)*
  United Kingdom
  44-1-294-1006
  SysOp: Andy Brimson
- SweFIG
  Per Alm Sweden
  46-8-71-35751
- NEXUS Servicios de Informacion, S. L.
  Travesera de Dalt, 104-106, Entlo. 4–5
  08024 Barcelona, Spain
  + 34 3 2103355 (voice)
  + 34 3 2147262 (modem)
  SysOps: Jesus Consuegra, Juanma Barranquero
  barran@nexus.nsi.es (preferred)
  barran@nsi.es
  barran (on BIX)

This list was accurate as of August 1990. If you know another on-line Forth resource, please let me know so it can be included in this list. I can be reached in the following ways:

Gary Smith
P. O. Drawer 7680
Little Rock, Arkansas 72217
Telephone: 501-227-7817
GEnie (co-SysOp, Forth RT and Unix RT): GARY-S
Usenet domain.: uunet!wugate!wuarchive!texbell!ark!lrark!gars

**ForthNet is a virtual Forth network that links designated message bases in an attempt to provide greater information distribution to the Forth users served. It is provided courtesy of the SysOps of its various*

(Continued from page 32.)

portable to other 83-Standard Forths, as long as the return-address-save sequences in `INIT-TEST` and `FINISH-TEST` are changed to save and restore only a single return stack element for most other Forths. Also, `TEST:` and `;DONE` should be redefined for use with other dictionary structures.

Interactive testing is important and useful (and, in fact, there is no reason why these tools cannot be used as an interactive testing format). However, once initial testing is done, it is often useful to have a permanent test suite in a consistent and readable format. Portions of many programs are so crucial to system operation that they merit a full validation suite to prove correct operation. At Harris, validation suites are being used on the instruction sets of some of the RTX processors. The tools presented here provide a starting point for creating a validation suite for a variety of applications.

Philip Koopman Jr. is a senior scientist at Harris Semiconductor and an adjunct professor at Carnegie Mellon University. The opinions in this article are his, and do not necessarily reflect the views of Harris Semiconductor.

(Continued from page 38.)

the current one. Malloc(), calloc(), talloc(), free() and friends all come down to brk() and sbrk() in the end. So there are "most primitive possible" functions. So primitive in fact that nobody in their right mind wants to use them if malloc() or something like it is available.

Note that, while this is true in Unix, it is not necessarily true in other operating systems. Consequently, while sbrk() is certainly the primitive memory allocation operation for Unix, it does not necessarily even *exist* on all C implementations. In particular, I would expect that it would be difficult to properly implement sbrk() on the Amiga (probably the Amiga C library simulates it with some restrictions). sbrk() assumes that each process has its own address space, which is not generally true. Use of sbrk() is not necessarily portable.

By the way, since brk() can be implemented in terms of sbrk(), sbrk() is the true primitive on Unix systems. In many Unix implementations, sbrk() is the true system call, and brk() is implemented as a library routine, a thin veneer around sbrk().
—Mitch Bradley

To suggest an interesting on-line guest, leave e-mail posted to GARY-S on GEnie (gars on Wetware and the Well), or mail me a note. I encourage anyone with a message to share to contact me via the above or through the offices of the Forth Interest Group.

FORTH DIMENSIONS

EDITORIAL

If you haven't paid close attention to the growth of on-line Forth activity, you may be surprised. Forth programs, debates, questions, news, and insights are being shared between several BBSs and larger communication systems—including some international ones—thanks to their respective sysops and to both electronic and manual gateways between systems. There is more reason than ever to tune in to the on-line Forth community. *FD*'s "Reference Section" lists the electronic resources we find and, despite some past problems, we try to keep it both current and complete. (You can help by informing us of changes and additions.)

If you didn't log on in August, you missed meetings scheduled with Bill Ragsdale and Glen Haydon. To further encourage your virtual presence on at least one of these electronic venues, upcoming guest conferences on GEnie's Forth RoundTable include:

Dick Miller, President of Miller Microcomputer Services
"To DOS or Not to DOS"
Thursday, September 20
9:30 p.m. Eastern/6:30 p.m. Pacific

Jef Raskin, originator of the Apple Mac and the Canon Cat
"What Happened to the Cat?"
Wednesday, October 17
9:30 p.m. Eastern/6:30 p.m. Pacific

(Note that the October conference is on Wednesday instead of the usual Thursday.)

* * *

Speaking of Glen Haydon (of MVP-FORTH, WISC, etc.), he has completed a significant revision of his book *All About Forth*. It has long been popular as the working reference volume of definitions, implementation examples, and relevant details about a widely used set of Forth words. But the recent, greatly revised and expanded version makes the book an annotated glossary of practically all Forth words in common usage, in all the primary dialects. Implementation examples are given in high-level Forth or 8086/88 assembly language to help clarify the text of a word's definition. When in doubt, just look it up! This essentially new book is, in my opinion, an important contribution to every Forth programmer's workbench. Look for it on the FIG Mail Order Form.

If you live in Memphis, don't blame us...

Publishing News reported that seventy-five percent of monthly magazines were delivered late in early 1990, an increase over last year. Memphis, Tennessee had the worst record (none delivered on time) and San Mateo, California had the best record (one hundred percent).

—Marlin Ouverson
Editor

Forth Dimensions
Published by the
Forth Interest Group
Volume XII, Number 3
September/October 1990
Editor
Marlin Ouverson
Advertising Manager
Kent Safford
Design and Production
Berglund Graphics

Forth Dimensions welcomes editorial material, letters to the editor, and comments from its readers. No responsibility is assumed for accuracy of submissions.

Subscription to *Forth Dimensions* is included with membership in the Forth Interest Group at $30 per year ($42 overseas air). For membership, change of address, and to submit items for publication, the address is: Forth Interest Group, P.O. Box 8231, San Jose, California 95155. Administrative offices and advertising sales: 408-277-0668.

About the Forth Interest Group

The Forth Interest Group is the association of programmers, managers, and engineers who create practical, Forth-based solutions to real-world needs. Many research hardware and software designs that will advance the general state of the art. FIG provides a climate of intellectual exchange and benefits intended to assist each of its members. Publications, conferences, seminars, telecommunications, and area chapter meetings are among its activities.

"*Forth Dimensions* (ISSN 0884-0822) is published bimonthly for $24/36 per year by the Forth Interest Group, 1330 S. Bascom Ave., Suite D, San Jose, CA 95128. Second-class postage paid at San Jose, CA. POSTMASTER: Send address changes to *Forth Dimensions*, P.O. Box 8231, San Jose, CA 95155."