VOLUME IX, NUMBER 2 JULY/AUGUST 1987 $5.00

FORTH DIMENSIONS

FORGETTABLE INTERNAL NAMES

STARFLIGHT, STAR BRIGHT

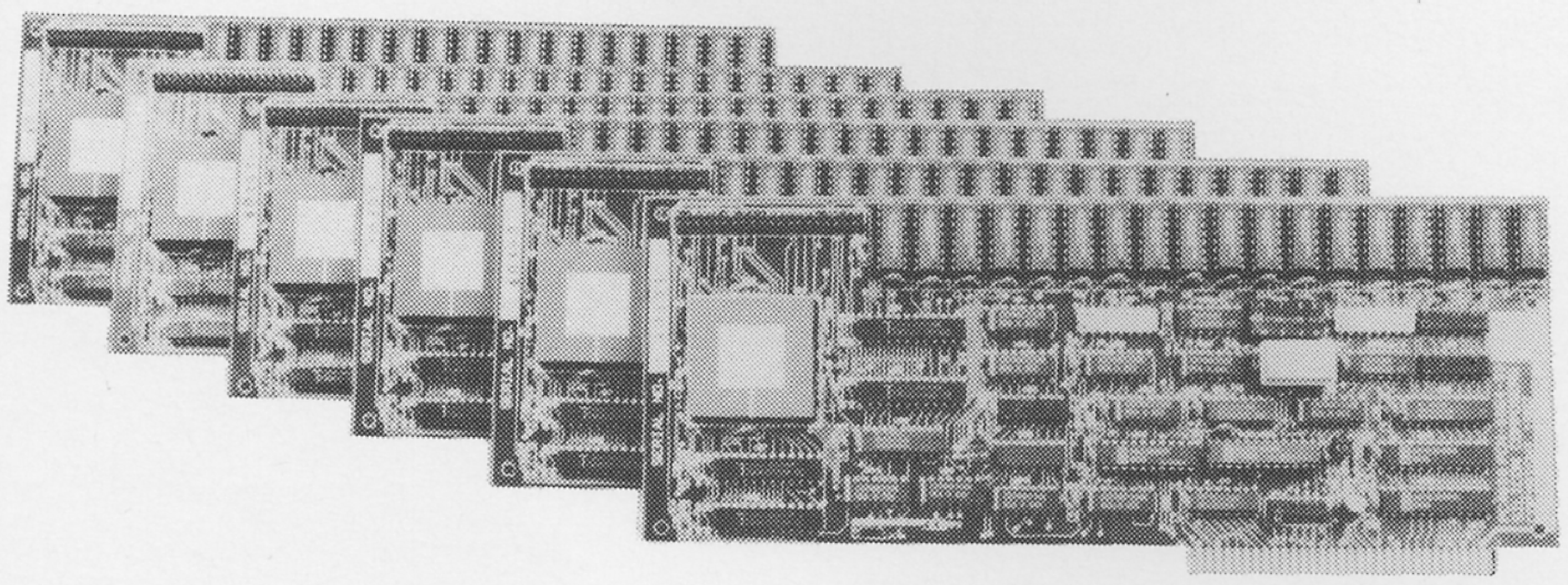
FLEXIBLE TEST ENVIRONMENT

EXECUTION SECURITY

INTRODUCING
THE PC-RISC SYSTEM

For High-Speed Concurrent Parallel Processing

ADD UP TO 30 MIPS TO YOUR PC!



THE PC-RISC SYSTEM

The PC-RISC System can turn your PC into a Forth "mini-supercomputer." This high speed parallel processing system can be configured using your PC, XT, or AT as the host by adding up to six PC4000s, an add-in card with the NC4000 Forth engine on board. PCX operating and development software is included. The PC-RISC System runs SCForth, cmForth, and Delta-C. PC4000/4MIPS $1295 each. PC4000/5MIPS $1495 each. OEM pricing available.

The PC-RISC System, Delta Board, and Delta Development System come fully assembled and tested with a 90-day warranty, communication software, user manual, a Forth development system, demo programs, and customer support using the Silicon Composers Bulletin Board (415) 322-2891.

SILICON COMPOSERS
210 California Avenue, Suite I
Palo Alto, CA 94306
(415) 322-8763

Formerly
SOFTWARE COMPOSERS

SILICON COMPOSERS

FORTH
DIMENSIONS

EDITORIAL

This year's National Forth Convention will be the occasion to celebrate ten years of the Forth Interest Group. C.H. Ting, the program director, points out that by understanding where Forth has come from, we can better see where it is going. The convention program promises to deliver presentations from people who have figured prominently in the history of the Forth Interest Group, who have made significant contributions to Forth's evolution, and who have witnessed the key turning points in our industry.

This will be a time to look both backward and forward in time, to assimilate the collective experiences of the past, and to gather ourselves for the requirements — and unexpected adventures — that the next ten years will bring. Look for details in this issue; we'll look for you in November.

As for looking forward, FIG's by-laws now provide for membership-wide election of its Board of Directors. Five positions will be filled by vote at the annual FIG meeting and election, to be held at the Forth National Convention this year (see "Candidates' Statements"). As the Forth Interest Group grows and becomes broader based, the full participation of its members is needed if the organization is to best serve the needs of those members. In colloquial terms: don't get frustrated, get involved. Another good reason to be at this year's convention.

Klaus Schleisiek has extended an invitation and call for papers to an upcoming euroFORML conference on "The Forth Programming Language and Forth Processors." From September 18-22, 1987 at Stettenfels Castle, West Germany (sponsored by Forth Gesellschaft eV, FRG), this year's conference will focus on Forth in hardware, and on the possibilities opened up by the quantum leap in speed of the new Forth processors.

Perched in the yicinity of Heilbronn (near Stuttgart), the castle holds 60 overnight guests, and 100 conferees. If you aren't geared up for a stay in the 12th century castle, reservations for more modern, bed-and-breakfast accommodations can be arranged. Conference languages will be English and, of course, Forth. Presentors can publish and orally present papers, hold special-interest "poster sessions," and conduct "on-demand" workshops. An advance deposit of DM200,- is required. Send camera-ready papers to Forth Gesellschaft by September 1; write C.D. Osten, Gneisenaustr. 23, D-2000 Hamburg 20, West Germany; or call (49) (40) 422 1694 or (49) (40) 490 5195. Klaus is available on Delphi as KS.

—*Marlin Ouverson*
Editor

"*Forth Dimensions* is published bi-monthly for $24/36 per year by the Forth Interest Group, 1330 S. Bascom Avenue, Suite D, San Jose, CA 95128. Second-class postage pending at San Jose, CA. 95101 POSTMASTER: Send address changes to the Forth Dimensions Group, P.O. Box 8231, San Jose, CA 95155."

Forth Dimensions
Published by the
Forth Interest Group
Volume IX, Number 2
July/August 1987
Editor
Marlin Ouverson
Advertising Manager
Kent Safford
Design and Production
Berglund Graphics
ISSN#0884-0822

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 ($43 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.

LETTERS

Sorting Out Batcher's

Dear Mr. Ouverson,

Many thanks for the article on "Batcher's Sort," by John Konopka (*FD* VIII/4). I am about the present this to my students in a Forth class here.

When typing **BSORT**, I found two offsetting errors. The program runs correctly, but gives a misleading impression about the role of **QQ** in **BSORT**. Consider the **BSORT** program fragment:

```
: BSORT  ( n -- )   ...
   BEGIN QRD-SET QQ
      BEGIN INNER-LOOP
      Q-TEST UNTIL...   ;
```

Q-TEST implies establishing a flag for **UNTIL** to read. However, the written **Q-TEST** gives a zero flag only if **QQ** <> **PP**, otherwise no flag at all.

Secondly, constant **QQ** of the fragment compensates by providing a number that acts like a flag. By actual test, **QQ** here is always the maximum possible value (two or greater) for any number *n* that is two or greater.

Enclosed is a rewritten program in Forth-83 — rather than the original Forth-79 — that embodies the change of **Q-TEST** always supplying a flag.

Thanks for good articles like these, and please keep them coming.

Sincerely yours,
Allen Anway
Superior, Wisconsin

Dear Marlin:

An interesting thing happened on the way to **BSORT**. I keyed in the screens but, not wanting to load a random number generator (see the data.tst screen enclosed), I simply wrote **INIT-DATA** to fill the **DATA** array with memory contents of bytes zero through 100. Then I executed **BSORT**. But what is this? When I executed **LIST-DATA**, the data were only partially sorted!

In a sort of muddled attempt to find out what was going on, I executed **BSORT** again — and again. After a number of these, the list was properly sorted. Why did it require multiple passes?

The best I can figure it, **BSORT** won't sort the data into proper order on one pass if some of the data are duplicated. Listing the contents of the **DATA** array before sorting, I noted many duplications.

The key to this problem is in the word **COMPARE-AND-SWAP**, which calls **SWAP-DATA** if one item of data is greater than the other when the two comparison keys generated by **INNER-LOOP** are used to index into the **DATA** array. Envision a sequence of data that gets swapped because 330 is found to be larger than 225 (the second item), but another pass will be required to make the final swap, producing the order ... 330, 225, 225, 225 ....

The listings of the files batcher.blk and data.tst offer one way of fixing the problem. I decided to use variables instead of constants, as I don't think the time savings is significant in my application. I also used F83's **DEFER-IS** to handle the forward reference to **KC**.

The fix I finally decided on uses **SORT-FLAG** to hold a flag that forces **BSORT** to repeat its execution of the original loops until a pass is made that requires no data swaps. If it is known that the data to be sorted will not have duplications, there is no problem. Otherwise, the definition of **SWAP-DATA**, or whatever definition replaces it, must set **SORT-FLAG** "on" when it executes. Of course, a new outer loop for **BSORT** will be needed in that case. Notice that **BSORT** sets **SORT-FLAG OFF** on each pass of the new outer loop; **BSORT** may be left as it is and the phrase **SORT-FLAG ON** put into **SWAP-DATA**, or not, depending on whether it is needed.

I assumed a purpose (and a definition) for the word ******. I have seen it defined elsewhere to produce altogether different results, but this seemed to make sense, at least to me; so, a reminder to authors: please define your non-standard words, even if they are deemed to be fairly well known.

Sincerely,
Gene Thomas
Central Arkansas FIG Chapter
Little Rock, Arkansas

Forget Automatically

When you are editing definitions in a block, the load-test-edit cycle can be speeded up by automating the **FORGET**ting of the first word in a block. This can be done by typing
CREATE TASK

NGS FORTH

A FAST FORTH, OPTIMIZED FOR THE IBM PERSONAL COMPUTER AND MS-DOS COMPATIBLES.

STANDARD FEATURES INCLUDE:

- ●79 STANDARD
- ●DIRECT I/O ACCESS
- ●FULL ACCESS TO MS-DOS FILES AND FUNCTIONS
- ●ENVIRONMENT SAVE & LOAD
- ●MULTI-SEGMENTED FOR LARGE APPLICATIONS
- ●EXTENDED ADDRESSING
- ●MEMORY ALLOCATION CONFIGURABLE ON-LINE
- ●AUTO LOAD SCREEN BOOT
- ●LINE & SCREEN EDITORS
- ●DECOMPILER AND DEBUGGING AIDS
- ●8088 ASSEMBLER
- ●GRAPHICS & SOUND
- ●NGS ENHANCEMENTS
- ●DETAILED MANUAL
- ●INEXPENSIVE UPGRADES
- ●NGS USER NEWSLETTER

A COMPLETE FORTH DEVELOPMENT SYSTEM.

PRICES START AT $70

NEW◆HP-150 & HP-110 VERSIONS AVAILABLE

NEXT GENERATION SYSTEMS
P.O.BOX 2987
SANTA CLARA, CA. 95055
(408) 241-5909

```
      SCREEN # 080
   0 CONSTANT TT      0 CONSTANT RR
   0 CONSTANT DD      0 CONSTANT PP
   0 CONSTANT NN      0 CONSTANT QQ
              ' 2DROP CONSTANT KC

   : 2**N 1 SWAP SHIFT ;    ( n --- 2**n )

                  ( index1\index2 --- )
   : KEY_COMP KC EXECUTE ;

   : SEL_T    NN 15 0 DO DUP I 2**N <=
     IF DROP I LEAVE THEN LOOP 1- 14 MIN
   ( --- )       [ ' TT 2+ ] LITERAL ! ;
   ( --- )
   : INNER-LP NN DD - 0 DO I PP AND RR =
     IF I DUP DD + KEY_COMP THEN LOOP ;

   : Q-TEST QQ PP = DUP 0=    ( --- flag )
     IF QQ PP - [ ' DD 2+ ] LITERAL !
        QQ 2/   [ ' QQ 2+ ] LITERAL !
        PP      [ ' RR 2+ ] LITERAL !
     THEN ;

   -->       names TT etc from Knuth


     SCREEN # 081
  : QRD-SET                        ( --- )
    TT 2**N [ ' QQ 2+ ] LITERAL !
     0      [ ' RR 2+ ] LITERAL !
    PP      [ ' DD 2+ ] LITERAL ! ;

  : BSORT                        ( n --- )
    ( n )   [ ' NN 2+ ] LITERAL ! SEL_T
    TT 2**N [ ' PP 2+ ] LITERAL !
    BEGIN   QRD-SET
      BEGIN INNER-LP Q-TEST
      UNTIL PP 2/ DUP
            [ ' PP 2+ ] LITERAL ! 0=
    UNTIL ;
                                FORTH-83
  -->   Allen Anway, Superior, WI, 1-9-87
  Batcher's sort-------John Konopka
                       Mitaka Shi, Japan
  Forth Dimensions, Nov-Dec 1986, page 39
  Knuth, Art of Computer Programming,
  Vol 3, pp 111-122, Addison-Wesley 1973

  To use, define SWAPPER ( ind1\ind2 --- )
  later, to check and perhaps swap.  Then,
  ' SWAPPER ' KC 2+ !    ( n --- ) BSORT
```

at the start of an editing session, and writing
FORGET TEST CREATE TASK
at the top of the block. When editing of the block is complete, the references to **TASK** can be erased. Even the manual insertion and deletion of marker words like **TASK** can be avoided if **LOAD** and **-->** are redefined so that they do the job.

The new **LOAD** constructs a distinctive name for the block to be loaded, and looks for a word with that

```
    0
BSORT fixed.          F83 2.0.1                    Dec1986:gt
As presented by John Konopka in FD Vol 8/4,  BSORT will not
correctly sort on one pass if there are data-duplicates in the
list to be sorted.  The number of passes required depends on how
many duplications there are of the same number, and with how
many different numbers that occurs.   Here, a sort flag is used
to force BSORT to repeat untill a pass with no data swap is made



                     Gene Thomas
                     7705 Apache Rd.
                     Little Rock, AR 72205



              Central Arkansas Forth Interest Group
```

```
    1
\ BSORT                                                Dec1986:gt
DEFER KC       ' NOOP IS KC    VARIABLE NN    VARIABLE RR
VARIABLE QQ    VARIABLE DD     VARIABLE PP    VARIABLE TT
VARIABLE SORT-FLAG
: KEY-COMPARE  KC  ;

: 2**N  ( n -- n*n )  DUP *  ;
: SELECT-T  NN @ 15 0 DO DUP I 2**N <= IF DROP I LEAVE THEN LOOP
   1- 14 MIN TT !  ;
: INNER-LOOP  NN @ DD @ - 0 DO I PP @ AND RR @  =
   IF I DUP DD @ + KEY-COMPARE THEN  LOOP  ;
: Q-TEST  QQ @ PP @ <> IF QQ @ PP @ - DD ! QQ @ 2/ QQ !
   PP @ RR !  0 THEN  ;
: QRD-SET  TT @  2**N QQ ! RR OFF PP @ DD !  ;

-->
```

```
    2
\ BSORT                                                          Dec1986:gt


\ n --  n is number of items to sort, must be possitive
: BSORT  SORT-FLAG ON
   BEGIN  SORT-FLAG @ WHILE  DUP   SORT-FLAG OFF
          ( repeat while sort-flag is on )          CR ." XXX"
     NN ! SELECT-T  TT @ 2**N  PP !
     BEGIN  QRD-SET QQ @
       BEGIN  INNER-LOOP  Q-TEST  UNTIL
       PP @ 2/ DUP PP ! 0=
     UNTIL
   REPEAT DROP ;

FROM B:DATA.TST  1 LOAD
\ count executions of ." xxx" to see how many passes were made
```

```
    3
\S  BSORT                                                        Dec1986:gt



KEY-COMPARE  call the compare-and-swap definition thru KC
             ( ' compare-and-swap is kc )
2**N         square n
SELECT-T     set outer loop limit

INNER-LOOP   compare the keys, swap out-of-order-data

Q-TEST       test for end of middle loop

QRD-SET      setup indice for Q-test
```

```
    0
\S  BSORT test                                       Dec1986:gt


INIT-DATA      on my Kaypro I this ensures quite a few data
               duplications, some numbers occuring >5 times

SWAP-DATA      swap the data in DATA+nKey and DATA+mKey
               SET SORT-FLAG ON IF THIS IS CALLED


COMPARE-AND-SWAP  compare the data pointed to by the keys output
                  by BSORT's inner-loop, swap if data+n is >
                  data+m

LIST-DATA      list DATA array
```

```
    1
\ BSORT test                                                  Dec1986:gt
VARIABLE X1    VARIABLE X2    CREATE DATA 200 ALLOT
: INIT-DATA  ( fill data array with memory contents )
    100 0 DO  I @  I 2* DATA + !  LOOP  ;

: SWAP-DATA  SORT-FLAG ON
           X1 @ DATA + @  X2 @ DATA + @
           X1 @ DATA + !  X2 @ DATA + !   ;

: COMPARE-AND-SWAP  ( n m -- )
    2* X1 !   2* X2 !   X1 @ DATA + @   X2 @ DATA + @  >
    IF  SWAP-DATA  THEN  ;
' COMPARE-AND-SWAP IS KC

: LIST-DATA  CR  100 0 DO  I 2* DATA + @  7 .R   I 1+  10 MOD 0=
    IF  CR  THEN   LOOP  ;
```

name in the dictionary. If it is found, **FORGET** is used to delete it and its successors. Then the name is used to **CREATE** a new null word, and the block is loaded in the usual way.

The definitions given here assume that the terminal input buffer can be temporarily relocated. This would not be necessary in a version of Forth in which **FORGET** and **CREATE** can be factored into two parts: a string-grabbing part and a part that refers to the address of the string.

To recover the space occupied by all the null words created by the new **LOAD**, **FORGET** the new **LOAD** and reload everything with the old **LOAD**.

Philip Bacon
Gainesville, Florida

Swift 6502 Multi-Tasking

Editor,

Richard Rooney wrote to you (*FD* VIII/6) asking whether anyone had implemented the Laxen model multitasker on a 6502. Yes.

Six months ago I, too, worked through the Laxen tutorial. Of course, the crux of the matter is getting around the fixed return stack in page one, and the data stack in page zero (fig-FORTH). My solution, after failure with block moves (too slow), was to add a little hardware. The "little hardware" is a stack segmentation scheme. Use a PIA to switch in a new page zero/one pair for the next task.

Overhead? Two additional assembly instructions. Very fast. This is very easy to do on the old OSI Challenger III system, because of the presence of pseudo-address lines A16-A19, which are PIA controllable. I had to modify an extra memory board, which became *stack memory*. This took only two ICs and about 25 jumpers. As Richard Rooney guessed, you use the **BRK** in place of Laxen's **RST**. My 6502 fig-FORTH multi-tasker handles 16 tasks. And it's fast.

I found Laxen's article to be very rewarding. So get those back issues and work through it!

Best regards,
Dale King
Leonard, Texas

Searching for F83

Dear FIG:

When I received the November/December issue of *Forth Dimensions*, I was writing a full-screen, memory-mapped editor for Laxen and Perry's F83. I had just gotten to the point where I was able to use my new editor to add the bells and whistles — one of which was the search function. I had not intended to rewrite the one provided, but Mr. Bill Zimmerly had submitted a fast, assembly language version of **SEARCH**. This got me interested, so I studied it with the intention of replacing the high-level **SEARCH** provided in F83.

I wish to address three points: 1. Brother Zimmerly asks, in the shadow screen accompanying his code, "How could it be faster?" I can make it faster (and smaller, too). 2. Mr. Zimmerly's code does not return the same results as the **SEARCH** provided in F83 — he returns an address and a flag, while F83 returns an offset and a flag. (3) Mr. Zimmerly's **SEARCH** is not case sensitive/neutral, as is Laxen and Perry's version.

Screen 15 is Mr. Zimmerly's, and compiles to 87 bytes, including headers for **(FIND1)** and **SEARCH**. If meta-compiled so that **(FIND1)** has no header in the new system, this can be reduced to 75 bytes.

Screen 16 is my modification of Mr. Zimmerly's **SEARCH**. It compiles to 67 bytes, and I claim it will run faster. First, I eliminated the **PUSH** and **POP** of the BP register, which serve no purpose since BP is not used and, therefore, is not changed. This saves four bytes. Next, I moved the exit code from a labeled routine into the **SEARCH** code, eliminating one jump for each execution of **SEARCH**. Finally, I moved the jump from the end of the routine to **HERE** (top of loop) in the **IF** statement. This will eliminate a jump every time there is a match on the first character but failure to match on the whole string.

(Continued on page 36.)

FORTH-79

FLEXIBLE TEST ENVIRONMENT

JOHN MULLEN - AMES, IOWA

Although Forth's interactive environment simplifies testing and debugging, there are times when a word or component is difficult to verify. Test prints can be used to clarify what is happening within a word at run time; but it is tedious to insert these print statements when they are needed, and then to either delete or comment them out when they are not.

The words described below set up a test environment that allows one to choose whether or not messages will be printed at run time, without modifying source screens. I have implemented this environment in polyFORTH on an IBM-AT, MVP-FORTH on a Texas Instruments Professional, and SuperFORTH on a Commodore-64. This article describes the environment, how to implement it, and how it may be used.

Test Mode Control

The first step is to establish a means of controlling the test environment. Screen 10 defines the constant **TEST?** which indicates test status. It is defined as a constant rather than as a variable, since it is to be used far more often that it is to be changed. The constants **YES**, **NO**, **TRUE**, and **FALSE** allow some variation in phrasing commands. The word **.L** will display the value of a logical flag as either YES or NO, so the effect of **TESTING?** is:

```
TESTING?  YES OK
```

The word **TESTING** will change the value of **TEST?**. For example, the phrase **YES TESTING** will set the testing mode.

Conditional Display Words

The test words defined in screen 11 will display information if the test mode is set, but will have no effect if it is not. The words **T.**, **TCR**, and **T.S** cover situations I encounter frequently. Their definition simply makes subsequent words more compact and easier to read. However, **TEST"** is a bit more complicated than the others.

Certainly, one could set up conditional prints as needed, for example:

```
... TEST? IF ." Two Real Roots " THEN ...
```

The message would appear only if the test mode is set, but this takes up more room in each definition, and is far less elegant, than:

```
... TEST" Two Real Roots " ...
```

The effect of **TEST"** is much like that of **ABORT"**, except that **ABORT** is not called. Its definition was adapted from that of **ABORT"** and **."** as described in *All About Forth* (Glen B. Haydon, Mountain View Press). Basically, two words are needed: **<TEST">** to conditionally print a string compiled into a word's definition, and **TEST"** to compile that string.

The immediate word **TEST"** will compile into the definition **<TEST">** and the string following **TEST"** in the input stream. For example, suppose the phrase below is entered:

```
... TEST" PHASE A " ...
```

After checking the state, **TEST"** compiles the code field address (CFA) of **<TEST">** into the definition. The ASCII code for " is 34, so the phrase **34 WORD** will place the string PHASE A into the dictionary. But **WORD** will not affect the dictionary pointer (**DP**), although it does leave its current value (**HERE**) on the stack. The phrase **C@ 1+ ALLOT** then increments the dictionary pointer to the first cell beyond the end of the string. This series of events is depicted in Figure One.

<TEST"> expects a string to be stored in the word's definition immediately after its address. Note that the string starts at the same address as that pushed on the return stack when **<TEST">** is called.

<TEST"> first sets up the string for **TYPE**, then adjusts the return address so that it points to the first address beyond the string. If this were not done, the interpreter would return to the count byte and first text byte of the string, with disastrous results. Then, if the test mode is set, **<TEST">** will display the compiled string; otherwise, the string's address and length count are dropped from the stack. This action is depicted in Figure Two.

The version of **<TEST">** shown has worked on all three Forth systems tested, but **TEST"** is somewhat system dependent, so it should be implemented with care. The version shown works in MVP-FORTH and SuperFORTH-64, but the string is compiled in the polyFORTH version of **TEST"** with the phrase **34 STRING**, which replaces lines 11 and 12 of screen 11.

If you want to try this in some other version of Forth, be aware that **WORD** does not work the same in all dialects. Some versions leave **HERE** on the stack and others do not. In addition, watch that **TEST"** and **<TEST">** agree on where the next address after the string is stored.

A lot of my personal taste goes into screens 10 and 11. There are many synonyms for **TRUE** and **FALSE**, and you may wish to add to or delete from the list of test words. But there are three suggestions I would make. First of all, set up **.L** to print the same number of characters in both the true and false case. This makes it easier to use **.L** to set up tables that look nice. Secondly, be sure that all **TEST** words are stack neutral. Note, for example, that **T.** duplicates the top value before printing. The final suggestion is to set up **TEST?** as a constant. Not only will you save two bytes in every defining screen and two in each definition, but you won't have to remember whether to use **@** or **C@**.

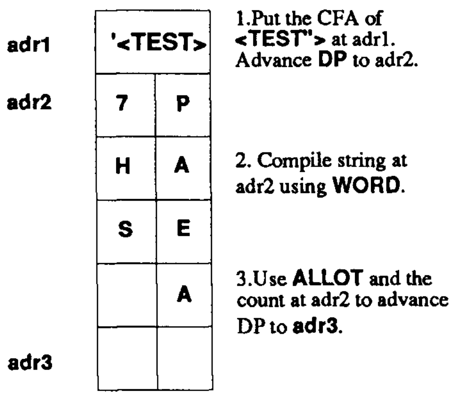

adr1
'<TEST>
1.Put the CFA of <TEST"> at adr1. Advance DP to adr2.
adr2
7
P
H
A
2. Compile string at adr2 using WORD.
S
E
A
3.Use ALLOT and the count at adr2 to advance DP to adr3.
adr3

Figure One. Action of **TEST"**

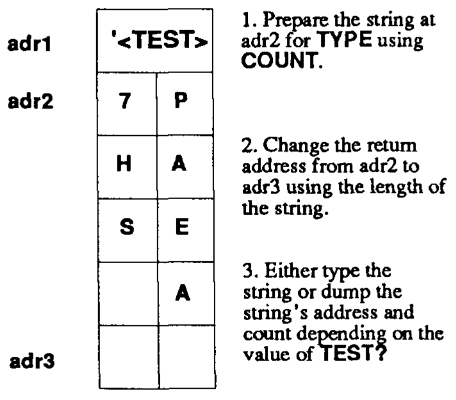

adr1
'<TEST>
1. Prepare the string at adr2 for TYPE using COUNT.
adr2
7
P
H
A
2. Change the return address from adr2 to adr3 using the length of the string.
S
E
A
3. Either type the string or dump the string's address and count depending on the value of TEST?
adr3

Figure Two. Action of **<TEST">**

```
SCREEN #9
( LOAD SCREEN FOR TEST ENVIRONMENT                      JPM 06NOV85)
10  LOAD
12 13 THRU
EXIT

  THE TEST ENVIRONMENT ALLOWS ONE TO DECIDE AT COMPILE TIME
WHETHER OR DOT DEBUGGING CODE SHOULD BE COMPILED INTO A WORD'S
DEFINTION.  IN ADDITION, THE DEBUGGING CODE IS CONDITIONAL SO
ONE CAN DECIDE WHETHER OR NOT TO EXECUTE THE CODE AT RUN TIME.


```

A Better Way

Nifty as this system is, there are two major drawbacks to its use. First of all, although nothing is displayed when the test mode is not set, **TEST?** is still being checked by each test word. **TEST"** is a real time consumer, since even when **TEST?** is false, **<TEST">** must still manipulate return addresses to avoid a system crash. The execution time for **TEST"** when **TEST?** is false is about 0.1 msec. on the IBM-AT, and 18.5 msec on the Commodore-64, which is enough time to affect some applications.

Secondly, when one is sure that a set of words no longer needs testing, its definitions are still cluttered up with the test prints, which uses dictionary space needlessly. Of course, you could go back and comment out all the test prints, but this is tedious and hard to change back, if an unexpected bug arises.

Screens 12 and 13 display more advanced versions of the words defined in screen 11. If **TEST?** is true at compile time, then these words work exactly as their earlier versions. If **TEST?** is false, however, nothing at all is compiled into the definition.

The method used for most of the words is straightforward. For example, **<TCR>** in screen 12 is defined exactly as **TCR** was in screen 10. Then, the immediate word **TCR** is defined to compile **<TCR>** into a word only if **TEST?** is true. This pattern can be followed for any of the other words, except **TEST"**.

The only problem with **TEST"** is how to deal with the string following it in the input stream if **TEST?** is false. The definition of **TEST"** in screen 13 functions just the same as the earlier version in screen 11 if **TEST?** is true, but uses **WORD** to move the interpreter beyond the string if **TEST?** is false. The polyFORTH version also uses **WORD** to do this, but has no **DROP** in line 10, since its version of **WORD** does not leave **HERE** on the stack.

Using the System

TEST? may be checked at both compile time and run time. If **TEST?** is true at compile time, conditional code is inserted so that you can turn displays on and off at run time. However, if **TEST?** is false, the code is not inserted and test displays will not appear at run time, regardless of the value of **TEST?**.

Suppose an application consists of five screens, four of which are thoroughly debugged, and one (screen 23) that is not. A load screen might appear, as follows:

```
NO  TESTING  22 LOAD
YES TESTING  23 LOAD
NO  TESTING  24 26 THRU
```

The result would be that any test code in screens 22, 24, 25, or 26 would not be compiled, but test code in screen 23 would. Thereafter, whenever **TEST?** is true, the test displays from screen 23 would appear.

```
SCREEN #10
( TEST ENVIRONMENT CONTROL                              JPM 06NOV85)
-1 CONSTANT TRUE     TRUE  CONSTANT YES
 0 CONSTANT FALSE    FALSE CONSTANT NO
 : .L    ( LF)  IF ." YES " ELSE ."  NO " THEN ;
YES CONSTANT TEST?
 : TESTING?    TEST? .L ;
 : TESTING    ( LF)  ['] TEST? ! ;    EXIT
  .L DISPLAYS THE VALUE OF A LOGICAL FLAG AS EITHER YES OR NO.
IT IS SET UP TO PRODUCE THE SAME WIDTH OF OUTPUT IN EITHER CASE
IN ORDER TO SIMPLIFY SETTING UP TABULAR DISPLAYS.
TEST? IS USED TO SIGNAL WHETHER OR NOT THE TEST MODE IS SET.
TESTING? WILL REPORT THE CURRENT STATE, TESTING WILL CHANGE IT.
E.G.   NO TESTING  RESETS THE TEST MODE.




SCREEN #11
( TEST" ETC, V1 - RUN TIME CHECKING OF TEST?        JPM 06NOV85)
: TCR    TEST? IF CR THEN ;
: T.     TEST? IF DUP . THEN ;
: T.S    TEST? IF .S THEN ;
: <TEST">    ( CONDITIONAL PRINT OF A COMPILED STRING )
  R@ COUNT  ( SET UP THE STRING FOR TYPE )
  DUP 1+ R> + >R   ( ADJUST RETURN ADDRESS)
  TEST? IF TYPE  ELSE 2DROP  THEN ;
: TEST"   ( COMPILATION OF A STRING AND <TEST"> )
  ?COMP               ( COMPILE ONLY)
  COMPILE <TEST">    ( RUN-TIME CODE FOR TEST" )
  34 WORD            ( DELIMITER IS " )
  C@ 1+ ALLOT ;      ( ADVANCE HERE TO THE END OF THE STRING )
IMMEDIATE                EXIT



SCREEN #12
( SIMPLE TEST WORDS CHECK TEST? AT COMPILE TIME     JPM 06NOV85)
: <TCR>    TEST? IF CR  THEN ;
: TCR    ?COMP  TEST? IF COMPILE <TCR> THEN ; IMMEDIATE
: <T.>    TEST? IF  DUP . THEN ;
: T.     ?COMP  TEST? IF COMPILE <T.>  THEN ; IMMEDIATE
: <T.S>    TEST? IF .S THEN ;
: T.S    ?COMP  TEST? IF COMPILE <T.S> THEN ; IMMEDIATE     EXIT










SCREEN #13
( TEST", VII - CHECKS TEST? AT COMPILE TIME         JPM 04MAY86)
: <TEST">    ( CONDITIONAL PRINT OF A COMPILED STRING )
  R@ COUNT  ( SET UP THE STRING FOR TYPE )
  DUP 1+ R> + >R    ( ADJUST RETURN ADDRESS)
  TEST? IF TYPE  ELSE 2DROP  THEN ;
: TEST"    ( CONDITIONAL COMPILATION OF A STRING AND <TEST"> )
  ?COMP  TEST? IF
    COMPILE <TEST">  ( RUN-TIME CODE FOR TEST" )
    34 WORD  C@ 1+ ALLOT  ( COMPILE THE STRING )
  ELSE
    34 WORD DROP          ( DISCARD THE STRING )
  THEN ;   IMMEDIATE          EXIT




```

FORTHkit

5 Mips computer kit

$400

Includes:

Novix NC4000 micro
160x100mm Fk3 board
Press-fit sockets
2 4K PROMs

Instructions:

Easy assembly
cmFORTH listing
shadows
Application Notes
Brodie on NC4000

You provide:

6 Static RAMs
4 or 5 MHz oscillator
Misc. parts
250mA @ 5V
Serial line to host

Supports:

8 Pin/socket slots
Eurocard connector
Floppy, printer,
video I/O
272K on-board memory
Maxim RS-232 chip

Inquire:

Chuck Moore's

Computer Cowboys

410 Star Hill Road
Woodside, CA 94062
(415) 851-4362

FORTH-83

FORGETTABLE INTERNAL NAMES

MICHAEL HORE - NUMBULWAR, AUSTRALIA

From time to time, there have been various proposals for dealing with internal, or local, names. By these, I mean the names of words that are only used internally in an application, never called from outside. A particularly nice scheme was presented by Dewey Val Schorre, back in *Forth Dimensions* II/5 ("Structured Programming by Adding Modules to Forth"). Dewey brought the notion of a "module," and presented three words (**INTERNAL**, **EXTERNAL**, and **MODULE**) to implement this notion. Each module is a self-contained unit, and communicates with the outside world by means of words defined between **EXTERNAL** and **MODULE** (as well as any constants or variables defined before **INTERNAL**). Words defined between **INTERNAL** and **EXTERNAL** may be referenced from within the module, but are not accessible from outside. Each module may be a complete application or a building block in a larger application.

Dewey's implementation of these words was simple but effective, involving the replacement of a dictionary link to remove the internal words from the search chain. Dewey acknowledges that such a simple implementation does not allow the dictionary space savings that could be possible — the headers of the internal words are unused after **MODULE** has been reached, but remain in the dictionary. (Note that the parameter fields of these definitions are needed for execution, but not the headers.)

Perhaps one reason why Dewey's words have not gained a wider acceptance is that, since his article, more elaborate vocabulary structures have evolved, and the effect of Dewey's words can be obtained by using vocabularies in an appropriate way. I have always felt, though, that Dewey's words state more clearly what is going on, since vocabularies have many other uses besides this particular one.

At this point, I am not going to attempt a full justification of a modular style of programming, which is now accepted as an essential discipline in the construction of any significant application. Let me just say that I have found it very useful to have this discipline enforced (the temptation to lapse into spaghetti code is always there). If I try to call an internal name from outside a module, it is good to have the system throw it back in my face. It means I probably have not understood either the problem or my "solution" to it! It also means that if I want to change the specification of an internal word, I know I don't have to look very far to find all the references to it.

Let us now turn to the question of the potential space savings. Note that we are not realizing this with modules implemented by means of vocabularies, any more than we could with Dewey's implementation of his words. That is the reason for the code presented here. I have found that space savings on the order of 20% is possible — the shorter the definitions, the greater the savings. This means that good Forth programmers will benefit more from this code than bad ones (bad programmers, please stop reading). This code will do even more — it will allow a whole module, parameter fields and all, to be loaded temporarily and then dismissed from memory, while leaving subsequent definitions intact. It is very useful to be able to do this with an assembler, as one obvious example.

It turns out that Dewey's three words are more suitable here than if we tried to do the job with vocabularies. That would not be impossible, just more difficult. Moreover, Dewey's words are very clear and say exactly what they mean — unlike, say, **FORGET-VOC**. Perhaps they may find a new lease on life with this implementation.

Working in this area inevitably brings us to various implementation dependencies, since the Forth standard quite rightly leaves unspecified such details as the internal structure of the dictionary. So what we will do is focus on one specific implementation model, Laxen and Perry's F83. This code should be easily adaptable to any Forth-83 system, however, and probably to many other Forths as well. On screen 67 we give the definitions of most of the non-standard F83 words we use, in case you need them. Unfortunately, there are several other words for which we can't do this without getting bogged down in a mass of irrelevant detail. But we will now try to give enough information for our purposes here.

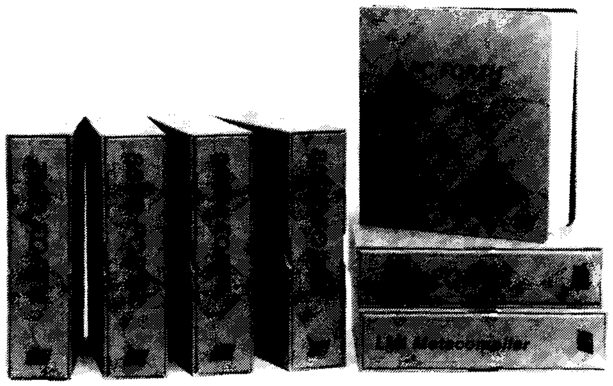
Firstly, the words **DEFER** and **IS**, respectively, create and redirect an execution vector. These words have been fully described by Henry Laxen in two "Techniques Tutorials" (*FD* III/6, V/6) that are well worth reading. Secondly, a couple of non-standard words are called from **FIND**, which we redefine on screen 72. However, the change we make to the definition of this word is very minor, and comes right at the end. Thus, we can ignore the details of the workings of **FIND**. In fact, even if you don't have F83, the change to **FIND** will probably be exactly the same. Thirdly, we need to refer here to the F83 word **HEADER**. The change is simply to rename it **(HEADER)**, so again we don't need to go into the details of its workings. We have more to say about it below.

Our scheme operates by separating the headers of the internal words. These go into a special area, the separated heads area (SH, for short) and are later forgotten. This is not as simple as it sounds, since internal words will have external words compiled after them, at which time the internal names must still exist. Later, if we forget the internal names by simply chopping the dictionary, we will lose the external names, too. Rather, we must follow the dictionary links and unlink any internal names, leaving the rest of the dictionary intact. We provide here a new forgetting word to do this — the regular **FORGET** need not be changed.

Now we will go into a bit more detail. The SH area has its own "dictionary pointer," the variable **SH-DP**. The base and top of the area are pointed to by two other variables, **SH-DP0** and **SH-TOP**. I am defining these as system variables here, although you may want to make them user variables if you do any multitasking on your system.

Each header in the SH area looks exactly like a normal header. The difference is that, instead of being followed by a code field, it is followed by a pointer to the code field. The code field itself sits in its usual position in the dictionary (and, of course, has no header in front of it). Here is how we set up this kind of header. In F83, headers are laid down by the word **HEADER**, which returns with the dictionary pointer **DP** pointing to where the code field will go. **CREATE** calls **HEADER**, then stores the code field. Our modification to **HEADER** is simply to make it vectored.

TOTAL CONTROL
with *LMI FORTH*™



For Programming Professionals:
an expanding family of compatible, high-performance, Forth-83 Standard compilers for microcomputers

For Development:
Interactive Forth-83 Interpreter/Compilers
- 16-bit and 32-bit implementations
- Full screen editor and assembler
- Uses standard operating system files
- 400 page manual written in plain English
- Options include software floating point, arithmetic coprocessor support, symbolic debugger, native code compilers, and graphics support

For Applications: Forth-83 Metacompiler
- Unique table-driven multi-pass Forth compiler
- Compiles compact ROMable or disk-based applications
- Excellent error handling
- Produces headerless code, compiles from intermediate states, and performs conditional compilation
- Cross-compiles to 8080, Z-80, 8086, 68000, 6502, 8051, 8096, 1802, and 6303
- No license fee or royalty for compiled applications

For Speed: CForth Application Compiler
- Translates "high-level" Forth into in-line, optimized machine code
- Can generate ROMable code

Support Services for registered users:
- Technical Assistance Hotline
- Periodic newsletters and low-cost updates
- Bulletin Board System

Call or write for detailed product information and prices. Consulting and Educational Services available by special arrangement.

LMI ***Laboratory Microsystems Incorporated***
Post Office Box 10430, Marina del Rey, CA 90295
Phone credit card orders to: ***(213) 306-7412***

Overseas Distributors.
Germany: Forth-Systeme Angelika Flesch, Titisee-Neustadt, 7651-1665
UK: System Science Ltd., London, 01-248 0962
France: Micro-Sigma S.A.R.L., Paris, (1) 42.65.95.16
Japan: Southern Pacific Ltd., Yokohama, 045-314-9514
Australia: Wave-onic Associates, Wilson, W.A., (09) 451-2946

So rename **HEADER** to **(HEADER)** and follow it with the code:

```
DEFER HEADER'
(HEADER) IS HEADER
```

When internal words are being compiler, **HEADER** is redirected to execute **SEP-HEADER**, which lays down a separated header and pointer to the code field. As required, we leave **DP** pointing to where the code field will go. **SEP-HEADER** is defined on screen 67. Notice that this word calls **(HEADER)** to do most of the work.

The F83 word **NAME>** takes the address of a name field and returns the corresponding compilation address (in F83 and most Forths, this is the same as the address of the code field). This word must be redefined so that, if the name is in the SH area, the pointer following the name is used to obtain the compilation address. This we do on screen 66. Other Forths should use either this word or something very similar. Any other words that go from a field within a header to the corresponding code field or parameter field will also need to be checked. If they use **NAME>** (or its equivalent) to make the transition, there's no problem. In F83, the only word in this category that doesn't use **NAME>** is **FIND**. As we mentioned above, the necessary change to this word is very minor. For completeness, we give the whole definition on screen 72.

This brings us to the new forgetting word. One of its features is that it takes a pair of addresses and forgets everything between them. So we will call it **<FORGET>** (pronounced "forget between"). Its definition is given on screen 68. Note the use it makes of the two other words on that screen, **UNLINK** and **TRIM**. **TRIM**, in its turn, uses **UNLINK** — it is **UNLINK** that does all the hard work.

F83 has a multi-thread dictionary structure, with the constant **#THREADS** defining the number of threads. **TRIM** contains a **DO** loop over the threads, calling **UNLINK** for each one. If your system does not have a multi-thread dictionary, removing the **DO** loop from **TRIM** may be sufficient.

<FORGET> can take a few seconds to execute, depending on your processor. This may not matter, as it won't be used with high frequency. However, if you can rewrite **UNLINK** in code for your particular processor, so much the better. This may save a few bytes of memory as well.

If you want to, you can replace the regular F83 forgetting scheme with this new one. (Thus, our **TRIM** has the same name as its F83 counterpart.) Simply redefine

```
: (FORGET)  DUP -1
 <FORGET>  DP !  ;
```

and replace **TRIM** with our new one. **FORGET** itself need not be altered. The advantage of doing this replacement is to save space, since our forgetting scheme is really an extension of the old one and covers much of the same ground. It works identically if there are no SH names, and the high limit is set to the highest address. If **UNLINK** is written in code, **FORGET** should run as fast as before.

The word **SET_SH_AREA** (screen 67) sets up the SH area by initializing the three pointers to this area. As written, it sets up 2000 bytes for the area in high memory, and 200 bytes below the current top of the parameter stack (which grows downwards). If this is not suitable for your system, change it as necessary. The number 2000, set up as the constant **SH_AREA_SIZE**, I found to be sufficient for my needs. If you have a big application, you may need to increase this number. The variable **SH-MAX** keeps track of the maximum size required so far for the SH area, so you can find by experimentation what size you need. Separate modules (i.e., not nested) reuse the same space in the SH area, to minimize the required size.

Modules may be nested in the fashion:

```
INTERNAL ...
INTERNAL...EXTERNAL...
 MODULE
EXTERNAL ...
MODULE
```

This is the same as provided in Dewey's original implementation of these words. Notice that once we are **EXTERNAL**, we can't become **INTERNAL** again in the same module. This could, however, be implemented. Being such perceptive readers, you will no doubt have noticed that **<FORGET>** is more general than we really need. Internal names to be forgotten will always occur in a single, contiguous cluster, whereas our implementation of **<FORGET>** will allow them to occur randomly, anywhere in the dictionary search order. This was, in fact, easier and shorter to implement (although slower). But it does mean that we can implement the more complicated module format, if necessary, without much trouble. I haven't done it here, feeling there may well be different schools of thought as to whether this feature will be desireable. Code is probably easier to follow if all **EXTERNAL** definitions are together, and this can usually be arranged quite easily. But go ahead and implement the more complicated format if you want to.

Our scheme for the temporary loading of entire modules (screen 71) is a straightforward application of **<FORGET>**. The word **TEMP_MODULE** saves **DP** on the stack, then resets it to halfway between where it was and the bottom of the SH area. You then load the temporary module, and use **END_TEMP** to set **DP** back to where it was. Hopefully, there is enough room between there and the temporary module for the definitions you now want to load. When this is done, **FORGET_TEMP** forgets the temporary module. If the temporary module has defined any vocabularies, remember to remove them from the search order before **FORGET_TEMP** — unless you find crashes entertaining! A very simple example is shown on screen 75.

Michael Hore is a missionary Bible translator with a programming background. He works among a remote group of Aborigines and uses an LSI 11/2. He is progressively moving all his software over to Forth, "...the way computers were meant to be programmed."

```
     65                                                          1065

 \ Separate header area                            May85 MRH   \ Separate header area                             May85 MRH
                                                               SH-DP0   points to the base of the SH area.
      VARIABLE  SH-DP0                                         SH-DP    Current SH area pointer.
      VARIABLE  SH-DP                                          SH-TOP   Points to the top of the SH area.
      VARIABLE  SH-TOP                                            Note: these three are all zero if the SH area has not been
 2000 CONSTANT  SH_AREA_SIZE                                      initialized.
      VARIABLE  EXTNL?                                         SH_AREA_SIZE  Size we allocate for SH area.  Alter if necessary.
      VARIABLE  SH-MAX                                         EXTNL?   A variable to allow us to check that the order
                                                                  EXTERNAL...MODULE is always followed (otherwise we'd crash).
                                                               SH-MAX      Records the maximum SH space used so far.



 : SH-HERE  ( -- addr )  SH-DP @  ;                            SH-HERE  Fetches the current SH area pointer.


     66                                                          1066

 \ SH area, cont.                                  May85 MRH   \ SH area, cont.                                   May85 MRH
 : WITHIN?  ( n lo hi -- n f )  OVER - >R  OVER SWAP -         WITHIN?  Is n within the range lo to hi inclusive?  My version
     R> SWAP U< NOT   ;                                           of this word has a couple of peculiarities - the test value
                                                                  is not popped, and the arithmetic is unsigned and circular,
                                                                  wrapping around from 64K to 0.
 : SEP_HDR?  ( addr -- addr f )  SH-DP0 @  SH-TOP @  WITHIN?  ;  SEP_HDR?  Is addr within the SH area?

 : ?SEP_HDR>  ( ?acf -- acf )  SEP_HDR?  IF  @  THEN   ;        ?SEP_HDR>  ?acf is either the address of a code field (acf),
                                                                  or a pointer to one if we are in the SH area.  Returns the
                                                                  acf.

 : NAME>  ( anf -- acf )                                       NAME>  Converts the addr of a name field (anf) to an acf.
     1 TRAVERSE 1+  ?SEP_HDR>   ;                                 TRAVERSE (F83) is the same as in Fig-FORTH.  Note that in F83
                                                                  link fields precede name fields, so that after the TRAVERSE
                                                                  we are normally looking at the code field.  At this point we
                                                                  insert ?SEP_HDR>.  This is the only change to the definition.

     67                                                          1067

 \ SH area, cont.                                  May85 MRH   \ SH area, cont.                                   May85 MRH

 : SEP_HEADER ( --(name) )  HERE ( save )  SH-HERE DP !        SEP_HEADER   Lays down a header in the SH area.
     (HEADER)   ( saved dp )  DUP ,   DP @  SH-DP !  DP !   ;

 : SET_SH_AREA   SP@ 200 -  DUP  SH_AREA_SIZE -                SET_SH_AREA   Sets up the SH area.  Alter this definition if
     DUP  HERE  200 +  U<  ABORT" Not enough room in memory"      necessary for your system.
     DUP  SH-DP0 !  SH-DP !  SH-TOP !   ;

 : ?SET_SH_AREA   SH-DP0 @ 0=  IF  SET_SH_AREA  THEN   ;       ?SET_SH_AREA  Sets up the SH area if not done already.

 \ 0  CONSTANT  FALSE
 \ -1 CONSTANT  TRUE                                           Here are some non-standard words included in F83.  Use these
 \ : ON   ( addr -- )  TRUE  SWAP !   ;                        definitions if you need to.
 \ : OFF  ( addr -- )  FALSE SWAP !   ;
 \ : ?PAIRS  ( n -- )  = NOT ABORT" Unbalanced structure"   ;
```

68

```
\ <FORGET>, etc.                                     May85 MRH
: UNLINK  ( lo hi 1st-link -- lo hi 1st-link )  DUP >R
   BEGIN   ?DUP
   WHILE   DUP >R  @
       BEGIN  2 PICK 2 PICK  WITHIN?  WHILE  @  REPEAT
       DUP  R> !
   REPEAT   R>   ;

: TRIM  ( lo hi voc-link-addr -- lo hi voc-link-addr )
   [ #THREADS 2* ] LITERAL  -
   #THREADS 0 DO   UNLINK  2+   LOOP   ;

: <FORGET>  ( lo hi -- )  OVER  FENCE @  U<  ABORT" Below FENCE"
   VOC-LINK  UNLINK
   BEGIN  @ ?DUP  WHILE  TRIM  REPEAT
   2DROP  ;
```

1068

```
\ <FORGET>, etc.                                     May85 MRH
UNLINK   Goes down a linked list starting with 1st-link,
   unlinking all links located at addresses within the range
   lo to hi (inclusive).  This word could profitably be put into
   code.
TRIM   For the given vocabulary, removes all words whose link
   fields are located within the range lo to hi.
   #THREADS is a constant giving the number of dictionary
   threads.
<FORGET>   Forgets all words whose link fields are located
   within the range lo to hi.  DP is not changed.
   We first check that lo is not below where FENCE points, to
   guard against wiping out the system.  Then we use UNLINK to
   remove any vocabularies that are to be forgotten.  (VOC-LINK
   is the head of the vocabulary list.)  Then we go down the
   pruned list of vocabularies and call TRIM for each one.
```

69

```
\ Modular programming words                          May85 MRH
: SAVE_HEADER  ( -- acf 20 )   ['] HEADER  >BODY @  20   ;

: INTERNAL    EXTNL? @
   ABORT" INTERNAL follows EXTERNAL - probably MODULE omitted"
   ?SET_SH_AREA   SH-DP @  ( save )   SAVE_HEADER
   200 SH-DP +!   ['] SEP_HEADER IS HEADER   ;

INTERNAL


: RESTORE_HEADER  ( old-hdr-acf 20 )  20 ?PAIRS   IS HEADER  ;

: (SH_FORGET)   ( faddr -- )
    DUP  SH-TOP @  <FORGET>   SH-DP !  ;
```

1069

```
\ Modular programming words                          May85 MRH
SAVE_HEADER   Saves the current setting of HEADER (DEFERred).

INTERNAL   Subsequent definitions will have separate headers.
   Note we reserve 200 bytes in the SH area for EXTERNAL names,
   in case this is a nested INTERNAL.


Now INTERNAL is defined, we can make use of it right away.


RESTORE_HEADER   Restores the previous setting of HEADER.

(SH_FORGET)   Forgets all (SH) names above the limit, faddr.
```

70

```
\ Modular programming words, cont.                   May85 MRH

: (EXTERNAL)  ( old-sh-dp old-hdr-acf 20 )    RESTORE_HEADER
    SH-HERE  SH-DP0 @  -  SH-MAX @  MAX  SH-MAX !
    SH-DP !   EXTNL? ON   ;

(EXTERNAL)

: EXTERNAL  (EXTERNAL)   ;



: MODULE   EXTNL? @ NOT
    ABORT" MODULE follows INTERNAL - EXTERNAL omitted"
    EXTNL? OFF  SH-HERE  (SH_FORGET)   ;
```

1070

```
\ Modular programming words, cont.                   May85 MRH

(EXTERNAL)  As for EXTERNAL below, but won't be accessible
   later, as an INTERNAL is now current.  Needs to be used
   to make us external again, so EXTERNAL itself will be
   accessible.


EXTERNAL   Subsequent defined names go where they were going
   before the last INTERNAL.  These names will still be
   accessible after MODULE.

MODULE   Forgets all names defined between INTERNAL and
   EXTERNAL.  Names after EXTERNAL are still accessible.
```

```
    71
\ TEMP_MODULE, etc.                                 May85 MRH

: TEMP_MODULE   ?SET_SH_AREA   HERE ( save )   SAVE_HEADER
   ['] (HEADER) IS HEADER
   HERE  SH-DP0 @  OVER -  2/ [ HEX ] 7FFE AND [ DECIMAL ]
   +  DP !   ;


: END_TEMP   RESTORE_HEADER   DP !   ;


: FORGET_TEMP   HERE  SH-DP0 @   (FORGET)   ;


MODULE
```

```
1071
\ TEMP_MODULE, etc.                                 May85 MRH

TEMP_MODULE  Marks the start of a module (such as the assembler)
   which is to be forgotten in toto once it has finished.
   This word leaves DP pointing to where the module will be
   loaded.  Currently this is half-way between HERE and the
   bottom of the SH area.  You can change this if necessary.

END_TEMP   Used after the temporary module is loaded.  Restores
   DP to its usual position.

FORGET_TEMP  Used when the temporary module is no longer needed.
   Forgets it using (FORGET), so nothing else is affected.
```

```
    72
\ Modified FIND                                     May85 MRH

: FIND   ( addr -- acf flag | addr false )
   PRIOR OFF   FALSE   #VOCS 0
   DO   DROP CONTEXT I 2* + @ DUP
      IF   DUP PRIOR @ OVER PRIOR !   =
         IF  DROP FALSE
         ELSE  OVER SWAP HASH @
               (FIND)  DUP ?LEAVE
   THEN THEN   LOOP
   DUP  IF  SWAP  ?SEP_HDR>  SWAP  THEN   ;
```

```
1072
\ Modified FIND                                     May85 MRH

The first part of this definition is copied straight from F83.








This extra line is the only change to the definition.
If the name was found, call ?SEP_HDR> to ensure we have the
address of the code field.
```

```
    75
\ Module example                                    Mar86 MRH

INTERNAL

: (CHANGE_CASE)  ( c -- c' )
   ASCII A  ASCII Z  WITHIN?  SWAP  ASCII a  ASCII z  WITHIN?
   ROT OR  IF  BL XOR  THEN  ;

EXTERNAL

: CHANGE_CASE  ( addr len -- )  BOUNDS
   DO  I C@ (CHANGE_CASE)  I C!  LOOP  ;

MODULE
```

```
1075
\ Module example                                    Mar86 MRH
This screen gives an example of a very simple module, which
provides a single word to the outside world.

(CHANGE_CASE)
   Changes the case of the given character, if it is
   alphabetic.

User word:

CHANGE_CASE
   Changes the case of the string (addr, len).
   BOUNDS (F83) is equivalent to OVER + SWAP .

Although CHANGE_CASE uses (CHANGE_CASE), the latter is now not
accessible, and its name is not taking up any memory space.
```

CONSUMERIZED FORTH

KEN TAKARA - SAN JOSE, CALIFORNIA

If you have ever played with Tinker Toys or Erector Sets, you'll have some appreciation for the fischertechnik products. In Europe, this West German company is extremely popular for its construction kits, ranging from simple block models similar to those by Lego, through sophisticated electronic, pneumatic, and hydraulic experimentation kits. The package we are concerned with is called the "fischertechnik computing" line, consisting, at the time of this writing, of a robotic construction kit.

With this kit, you can design and build simple, two-axis robots, connect them to your computer, then program and run them. The original kits from fischertechnik use BASIC as the programming language. When we (at Parsec Research) saw the kit, we were enthralled by the possibilities it suggested. Being Forth-oriented, we naturally couldn't help but question the choice of BASIC as a language for conducting experiments with robots. So we decided to create an "improvement" on the original. The result was PaRCL (pronounced "parkul"), Parsec Robot Control Language.

The original kits came with very little help by way of manuals. For the dedicated computer hobbyist, the program listings and minimal explanations would be adequate, but to the average consumer, the thing would be totally cryptic. Given this, I decided to write a text to help a novice computerist learn how to program the robots described in the kit. The primary consideration in designing PaRCL was this: I must be able to explain what is happening as simply as possible. If the explanation for some language feature is too cumbersome, then it probably is inappropriate to include it in the language.

Why Forth is Used

Most Forth programmers already know why their favorite language is superior to all others. Most Forth programmers are also aware of the fierce opposition that sometimes exists among others. Epithets aside, I selected Forth asthe basis of a consumer-oriented product for three reasons: functionality, interactivity, and familiarity.

Forth is a functional language; that is, when you program in Forth, you create functions to express an operation you wish to perform. Thus, you can actually create a collection of specialized vocabularies to talk about the different types of things you want your program to do. This makes it very easy for me to explain to someone how to make the robot do some sequence of actions.

For example, suppose I give this description for a robotic arm:

```
EXTEND-ARM-TO-TARGET
PICK-UP-OBJECT
RETRACT-ARM
ROTATE-TO-POSITION-3
EXTEND-ARM-TO-TARGET
DROP-OFF-OBJECT
```

This pretty much says it all. Naturally, we must assume here that I've already described what each of the steps entails, and that the reader is acquainted enough with Forth syntax to accept the hyphenated words as single commands.

Forth's interactivity is an immediate benefit. If I have taken the user through the steps necessary to define the words given in the above example, I can have him actually try them out immediately.

Finally, I am familiar with Forth. Given the choice, I generally prefer it to another language. Of course, given an appropriate incentive, I *might* consider learning COBOL...

Consumerization

When micros first became available, the language of choice (the *only* language, for that matter) was BASIC. Now, with Borland's efforts, it seems that everyone is learning Pascal. So how did I expect to foist yet another language (and a peculiar one, at that) on an unsuspecting public? Well, the first problem when getting someone to try something new, is to make it palatable (or at least marginally tolerable). Thus, it was necessary to "consumerize" Forth for this application.

As I stated earlier, when designing PaRCL, my primary criterion for inclusion of any language feature was explainability. If I could explain it, it was probably useable. If I could not explain it, then it was probably useless. This meant that a lot of standard Forth words were scrapped from the PaRCL vocabulary. After all, it is intended to be used to write *robot* programs, not *computer* programs.

Most Forth books start by showing the novice programmer how to display messages and numbers. Then they jump into a discussion of the parameter stack. Now, this is the second most frequently criticized fact of Forth life: the existence of a naked stack. (The first most criticized aspect is reverse polish.) To a Forth

CALL FOR PAPERS

for the ninth annual

FORML CONFERENCE

The original technical conference for professional Forth programmers, managers, vendors, and users.

Following Thanksgiving, November 27-29, 1987

Asilomar Conference Center
Monterey Peninsula overlooking the Pacific Ocean
Pacific Grove, California, USA

Theme: Forth and the 32-bit Computer

Computers with large address space and 32-bit architecture are now generally available at industrial and business sites. Forth has been installed and Forth applications programs are running on these computers. Graphic displays and applications are currently demanded by users. Implementation of Forth and meeting these requirements is a challenge for the Forth professional. Papers are invited that address relevant issues such as:

Large address spaces in 32-bit computers.
The graphic display, windows, & menu handling.
Relation to operating systems, other languages, & networks.
Control structures, data structures, objects, & strings.
Files, graphics, & floating point operations.
Comparison with 16-bit computers.

Papers on other Forth topics are also welcome. Mail your abstract(s) of 100 words or less by September 1, 1987 to:

FORML Conference
P. O. Box 8231
San Jose, CA 95155, USA

Completed papers are due November 1, 1987. For registration information call the Forth Interest Group business office at (408) 277-0668 or write to **FORML Conference.**

Asilomar is a wonderful place for a conference. It combines comfortable meeting and living accommodations with secluded forests on a Pacific Ocean beach. Registration includes deluxe rooms, all meals, and nightly wine and cheese parties.

programmer it is simply accepted, and often abused. To a novice, however, it can be a major roadblock. Since I was catering to novices, I decided to reduce the number of stumbling blocks by eliminating any mention of the stack. This meant that all our beloved stack words (**DUP**, **SWAP**, **ROT**, **DROP**, etc.) had to go.

Instead of the stack, I emphasized the use of variables. I realize the stack is inteded to improve certain aspects of Forth programming. But I also know from experience that most novice programmers feel much more comfortable with variables, which they simply accept, and often abuse. In the case of a systems programmer concerned with efficiency or speed, this is a problem. But for everyone else, who cares it it's slow, as long as it works, and the programmer's intention is clear?

Naturally, for this somewhat lobotomized version of Forth, most of the disk operators, dictionary operators, double-word arithmetic, and data-conversion words were eliminated as irrelevant for simple robot control. Of course, the user should have some way to write programs and to save them, so I included a source editor and a few disk operators such as **LOAD**, **THRU**, and **LIST**. I also included a hard-copy word, **PRINT-SCREENS**.

I did use Forth screens. It was easier to explain screens than to create a file system, and people seem to get a kick out of having unrestricted access to the disk. The business of opening files, appending to files, and closing files is complicated enough, and more so to a computer novice. Also, if I supported a file system, I would have to explain how to use it, and that would have been a chore. Being lazy, I took the easy way out.

Preserved Features

So, considering what was removed, you probably wonder that I left anything Forth-like intact. I did. In fact, PaRCL uses reverse Polish without any modification. Thinking in terms of passed parameters, it is relatively easy to explain how the output from one command (I called them commands instead of words) is held by the computer, and is used as input to another command. To a complete novice, the computer is quite mysterious anyway, so I took the liberty of simply using reverse Polish without any technical explanation.

This may seem contradictory, compared to my previous concern for explainability. How can I worry about explaining everything, then decline to explain the mysteries of reverse Polish? In fact, my intention was to explain how to use PaRCL to program the robots. Since the arithmetic for this kit is relatively simple, there was no reason to bother with algebraic parsing. Most people are willing to accept a few peculiarities, as long as they are kept to a minimum.

I also kept single-word integer arithmetic. Again, for this kit, this was entirely satisfactory. The physical limits of the components are well within single-precision boundaries.

Can Forth Succeed?

It has long been the dream of Forth programmers to bring their language to the attention of the masses. We would all like to see Forth accepted as a "respectable" language for programming. Will our attempt with PaRCL successfully engender acceptance of Forth among non-Forth users? What is required for success in this effort?

I don't know whether or not Forth will ever be popularly received. I certainly didn't choose it in order to persuade anyone that it is the best language; it was simply the easiest language for presenting robotic programming.

The acceptance of Forth outside the diehard Forth community depends on two factors: the manner in which it is presented, and the receptiveness of the audience. If you try to proselytize, you will probably face a great deal of resistance. If you present it to a novice programmer with promises of effortless programming, you will surely disappoint him. And if you make exaggerated claims about Forth, you will always provoke an argument.

An experienced Fortran programmer usually has no desire to learn another language; he is already at ease with what he has, and can do whatever he needs. The same may be said of a BASIC programmer, an Ada programmer, and so on. Half of these people see no point in struggling through the entire learning process with an unfamiliar language. Another quarter of them are actively hostile to anyone who might challenge the capabilities of their favorite language. And the last quarter, interested in programming languages in general, will pick up anything promising that comes their way.

Any programmer willing to learn and use Forth should be able to appreciate its merits (and its drawbacks) without further argument. For any other programmer, it is pointless to debate the issue, just as it is pointless to discuss the merits of PL/1 with a Forth devotee.

A casual computer user will probably never gain any insight or benefit from learning Forth (or any other language, for that matter). A person who uses computers frequently might, as long as it doesn't tax his patience every time he tries to write some simple program with it. For most computer users, to have to deal with the complexities of stacks, dictionary structures, linked lists, reverse Polish notation, virtual memory, *and* memory-mapped I/O in order to patch together a checkbook-balancing program (yes, they still do it) is an enormous bother.

As long as Forth is merely a dedicated hacker's language, it will remain merely a dedicated hacker's language. If it is to become a *user's* language, as popular as BASIC, it will have to put on some respectable clothes and make itself presentable to the public. And how might we do that? Well, you'll have to figure *that* out for yourself.

Ken Takara is technical director for Parsec Research (SuperFORTH 64) and writes the "Designers Debate" column for Computer Language. *Ken says he will probably continue to embarrass himself publicly by speaking at the West Coast Computer Faires.*

FIG
MAIL ORDER FORM

MEMBERSHIP IN THE FORTH INTEREST GROUP

109 - **MEMBERSHIP** in the FORTH INTEREST GROUP and Volume 9 of FORTH DIMENSIONS. No sales tax, handling fee, or discount on membership. See the back page of this order form.

The Forth Interest Group is a world-wide, non-profit, member-supported organization with over 4,000 members and 90 chapters. FIG membership includes a subscription to the bi-monthly publication, FORTH Dimensions. FIG also offers its members group health and life insurance, an on-line data base, a large selection of Forth literature and many other services. Cost is $30.00 per year for USA, Canada & Mexico; all other countries $42.00 per year. The annual membership dues are based on the membership year, which runs from May 1 to April 30.

When you join, you will receive issues that have already been circulated for the current volume of Forth Dimensions, and subsequent issues will be mailed to you as they are published. You will also receive a membership card and number.

■ *HOW TO USE THIS FORM*

1. Each item you wish to order lists three different price categories:

Column 1 - USA, Canada, Mexico
Column 2 - International Surface Mail
Column 3 - International Air Mail

2. Select the item and note your price in the space provided.

3. After completing your selections, enter your order on the fourth page of this form.

4. Detach the form and return it with your payment to the Forth Interest Group.

■ *FORTH DIMENSIONS BACK VOLUMES*

The six issues of the volume year (May — April)

101	— Vol. 1	FORTH Dimensions	(1979/80)	$15/16/18	______
102	— Vol. 2	FORTH Dimensions	(1980/81)	$15/16/18	______
103	— Vol. 3	FORTH Dimensions	(1981/82)	$15/16/18	______
104	— Vol. 4	FORTH Dimensions	(1982/83)	$15/16/18	______
105	— Vol. 5	FORTH Dimensions	(1983/84)	$15/16/18	______
106	— Vol. 6	FORTH Dimensions	(1984/85)	$15/16/18	______
107	— Vol. 7	FORTH Dimensions	(1985/86)	$20/21/24	______
108	— Vol. 8	FORTH Dimensions	(1986/87)	$20/21/24	______

■ *FORML CONFERENCE PROCEEDINGS*

FORML PROCEEDINGS — FORML (the Forth Modification Laboratory) is an informmal forum for sharing and discussing new or unproven proposals intended to benefit Forth. Proceedings are a compilation of papers and abstracts presented at the annual conference. FORML is part of the Forth Interest Group.

310 — FORML PROCEEDINGS 1980 $30/33/40 ________
Technical papers on the Forth language and extensions.

311— FORML PROCEEDINGS 1981 $45/48/55 ________
Nucleus layer, interactive layer, extensible layer, metacompilation, system development, file systems, other languages, other operating systems, applications and abstracts without papers.

312— FORML PROCEEDINGS 1982 $30/33/40 ________
Forth machine topics, implementation topics, vectored execution, system development, file systems and languages, applications.

313— FORML PROCEEDINGS 1983 $30/33/40________
Forth in hardware, Forth implementations, future strategy, programming techniques, arithmetic & floating point, file systems, coding conventions, functional programming applications.

314— FORML PROCEEDINGS 1984 $30/33/40______
Expert systems in Forth, using Forth, philosophy, implementing Forth systems, new directions for Forth, iterfacing Forth to operating systems, Forth systems techniques, adding local variables to Forth.

315— FORML PROCEEDINGS 1985 $35/38/45________
Also includes papers from the 1985 euroFORML Conference. Applications: expert systems, data collection, networks. Languages: LISP, LOGO, Prolog, BNF. Style: coding conventions, phrasing. Software Tools: decompilers, structure charts. Forth internals: Forth computers, floating point, interrupts, multitasking, error handling.

316— FORML PROCEEDINGS 1986 $30/33/40______ NEW
Forth internals, Methods, Standards, Forth processors, Artificial Intelligence, Applications.

■ *BOOKS ABOUT FORTH*

200 — ALL ABOUT FORTH $25/26/35 ________
Glen B. Haydon
An annotated glossary for MVP Forth; a 79-Standard Forth.

216 — DESIGNING & PROGRAMMING PERSONAL EXPERT SYSTEMS $19/20/29 ________
Carl Townsend and Dennis Feucht
Introductory explanation of AI-Expert System Concepts. Create your own expert system in Forth. Written in 83-Standard.

217 — F83 SOURCE $20/21/30________
Henry Laxen & Michael Perry
A complete listing of F83 including source and shadow screens. Includes introduction on getting started.
218 — FOOTSTEPS IN AN EMPTY VALLEY
(NC4000 Single Chip Forth Engine) $25/26/35 ________
Dr. C. H. Ting
A thorough examination and explanation of the NC4000 Forth chip including the complete source to cmForth from Charles Moore.
219 — FORTH: A TEXT AND REFERENCE $22/23/33 ________
Mahlon G. Kelly & Nicholas Spies
A textbook approach to Forth with comprehensive references to MMS-FORTH and the 79 and 83 Forth Standards.
220 — FORTH ENCYCLOPEDIA $25/26/35 ________
Mitch Derick & Linda Baker
A detailed look at each fig-FORTH instruction.
225 — FORTH FUNDAMENTALS, V.1 $16/17/20 ________
Kevin McCabe
A textbook approach to 79-Standard Forth
230 — FORTH FUNDAMENTALS, V.2 $13/14/18 ________
Kevin McCabe
A glossary.
232 — FORTH NOTEBOOK $25/26/35 ________
Dr. C. H. Ting
Good examples and applications. Great learning aid. PolyFORTH is the dialect used. Some conversion advice is included. Code is well documented.
233 — FORTH TOOLS $22/23/32 ________
Gary Feierbach & Paul Thomas
The standard tools required to create and debug Forth-based applications.
235 — INSIDE F-83 $25/26/35 ________
Dr. C. H. Ting
Invaluable for those using F-83.
237 — LIBRARY OF FORTH ROUTINES AND UTILITIES
James D. Terry $23/25/35 ________
Comprehensive collection of professional quality computer code for Forth; offers routines that can be put to use in almost any Forth application,. including expert systems and natural language interfaces.
240 — MASTERING FORTH $18/19/22________
Anita Anderson & Martin Tracy
A step-by-step tutorial including each of the commands of the Forth-83 International Standard; with utilities, extensions and numerous examples.
245 — STARTING FORTH, 2nd Edition (soft cover)
Leo Brodie $20/21/30 ____ NEW
In this new edition of Starting Forth, the most popular and complete introduction to Forth, syntax has been expanded to include the new Forth '83 Standard.
246 — STARTING FORTH (hard cover) $20/21/30 ________
Leo Brodie
255 — THINKING FORTH (soft cover) $16/17/20 ________
Leo Brodie
The sequel to "Starting Forth". An intermediate text on style and form.
265 — THREADED INTERPRETIVE LANGUAGES
R. G. Loelinger $25/26/35 ________
Step-by-step development of a non-standard Z-80 Forth.
267 — TOOLBOOK OF FORTH $23/25/35 ____ NEW
(Dr. Dobb's)
Edited by Marlin Ouverson
Expanded and revised versions of the best Forth articles collected in the pages of Dr. Dobb's Journal.

270 — UNDERSTANDING FORTH $3.50/5/6 ________
Joseph Reymann
A brief introduction to Forth and overview of its structure.

■ *ROCHESTER PROCEEDINGS*

The Institute for Applied Forth Research, Inc. is a non-profit organization which supports and promotes the application of Forth. It sponsors the annual Rochester Forth Conference.

321 — ROCHESTER 1981 $25/28/35 ________
(Standards Conference)
79-Standard, implementing Forth, data structures, vocabularies, applications and working group reports.
322 — ROCHESTER 1982 $25/28/35 ________
(Data bases & Process Control)
Machine independence, project management, data structures, mathematics and working group reports.
323 — ROCHESTER 1983 $25/28/35 ________
(Forth Applications)
Forth in robotics, graphics, high-speed data acquisition, real-time problems, file management, Forth-like languages, new techniques for implementing Forth and working group reports.
324— ROCHESTER 1984 $25/28/35 ______
(Forth Applications)
Forth in image analysis, operating systems, Forth chips, func-tional programming, real-time applications, cross-compilation, multi-tasking, new techniques and working group reports.
325 — ROCHESTER 1985 $20/21/30 ________
(Software Management & Engineering)
Improving software productivity, using Forth in a space shuttle experiment, automation of an airport, development of MAGIC/L, and a Forth-based business applications language; includes working group reports.

■ *THE JOURNAL OF FORTH APPLICATION & RESEARCH*

A refereed technical journal published by the Institute for Applied Forth Research, Inc.

401 — JOURNAL OF FORTH RESEARCH V.1
Robotics/Data Structures $30/33/38 ________
403 — JOURNAL OF FORTH RESEARCH V.2 #1
Forth Machines $15/16/18 ________
404 — JOURNAL OF FORTH RESEARCH V.2 #2
Real-Time Systems $15/16/18 ________
405 — JOURNAL OF FORTH RESEARCH V.2 #3
Enhancing Forth $15/16/18 ________
406 — JOURNAL OF FORTH RESEARCH V.2 #4
Extended Addressing $15/16/18 ________
407 — JOURNAL OF FORTH RESEARCH V.3 #1
Forth-based laboratory systems and data structures.
$15/16/18 ________
409 — JOURNAL OF FORTH RESEARCH V.3 #3
Application Languages $15/16/18 ________
410 — JOURNAL OF FORTH RESEARCH V.3 #4
Applications, Arthmatic extensions $15/16/18 ________

■ *DR. DOBB'S JOURNAL*

This magazine produces an annual special Forth issue which includes source-code listing for various Forth applications.
422 — DR. DOBB'S 9/82 $5/6/7 ________
423 — DR. DOBB'S 9/83 $5/6/7 ________
424 — DR. DOBB'S 9/84 $5/6/7________
425 — DR. DOBB'S 10/85 $5/6/7 ________

■ *HISTORICAL DOCUMENTS*

501 — KITT PEAK PRIMER $25/27/35________

One of the first institutional books on Forth. Of historical

502 — fig-FORTH INSTALLATION MANUAL $15/16/18 ________

Glossary model editor — we recommend you purchase this manual when purchasing the source code listing.

503 — USING FORTH $20/21/22 ________

FORTH, Inc.

■ *REFERENCE*

305 — FORTH 83-STANDARD $15/16/18________

The autoritative description of 83-Standard Forth. For reference, not instruction.

300 — FORTH 79-STANDARD $15/16/18 ________

The authoritative description of 79-Standard Forth. Of historical interest.

■ *REPRINTS*

420 — BYTE REPRINTS $5/6/7 ___________

Eleven Forth articles and letters to the editor that have appeared in Byte magazine.

■ *ASSEMBLY LANGUAGE SOURCE CODE LISTINGS*

Assembly Language Source Listings of fig-FORTH for specific CPUs and machines with compiler security and variable length names.

514 — 6502/SEPT 80	$15/16/18	________
515 — 6800/MAY 79	$15/16/18	________
516 — 6809/JUNE 80	$15/16/18	________
517 — 8080/SEPT 79	$15/16/18	________
518 — 8086/88/MARCH 81	$15/16/18	________
519 — 9900/MARCH 81	$15/16/18	________
521 — APPLE II/AUG 81	$15/16/18	________
523 — IBM-PC/MARCH 84	$15/16/18	________
526 — PDP-11/JAN 80	$15/16/18	________
527 — VAX/OCT 82	$15/16/18	________
528 — Z80/SEPT 82	$15/16/18	________

■ *MISCELLANEOUS*

601 — T-SHIRT SIZE __________ NEW

"May the Forth Be With You"

Small, Medium, Large and Extra-Large

White design on a dark blue shirt. $12/13/14 ________

602 — POSTER (BYTE Cover) $5/6/7 ________

616 — HANDY REFERENCE CARD FREE ________

683 — FORTH-83 HANDY REFERENCE CARD FREE ________

■ *FORTH MODEL LIBRARY*

The model applications disks below are the first releases of new professionally developed Forth applications. 5 1/4" disks are IBM MS-DOS 2.0 and up compatible and are compatible with Forth-83 systems listed below.

Laxen/Perry F83

LMI PC/FORTH 3.0

MasterFORTH 1.0

TaskFORTH 1.0

PolyFORTH (R) II

Macintosh 3 1/2" disks are available for MasterFORTH systems only.

Please specify disk size when ordering ____________________

701 — A FORTH LIST HANDLER V.1 $40/43/45 ________

by Martin J. Tracy

Forth is extended with list primitives to provide a flexible high-speed en-vironment for artificial intelligence. ELISA and Winston & Horn's micro-LISP are included as examples. Documentation is included on the disk.

702 — A FORTH SPREADSHEET V.2 $40/43/45 ________

by Craig A. Lindley

This model spreadsheet first appeared in Forth Dimensions Volume 7, Issue 1 and 2. Those issues contain the documentation for this disk.

703 — AUTOMATIC STRUCTURE CHARTS V.3 .

by Kim R. Harris $40/43/45 ________

These tools for the analysis of large Forth programs were first presented at the 1985 FORML conference. Program documentation is contained in the 1985 FORML Proceedings.

704 — A SIMPLE INFERENCE ENGINE V.4 $40/43/45 ______ NEW

by Martin J. Tracy

Based on the Inference Engine in Winstom & Horns book of *Lisp*, this volume takes you from pattern variables to a complete unification algorithm. Accompanied throughout with a running commentary on Forth philosophy and style.

706 — THE MATH BOX V.6 $40/43/45 ______ NEW

by Nathaniel Grossman

A collection of mathematical routines by the foremost author on math in Forth. Extended double precision arithmetic, a complete 32-bit, fixed-point math package and auto-ranging text graphics are included. There are utilities for rapid polynomial evaluation, continued fractions and Monte Carlo factorization.

■ *NC4000SERIES*

801 — MORE ON NC4000, VOLUME 1 $10/11/14 ______ NEW

FIG-Tree style forum on NC4000. Topics including bugs, products, tips, benchmarks, and NC4000 instruction bit patterns. Chuck Moore's teleconference. 2nd edition.

802 — MORE ON NC4000, VOLUME 2 $15/16/18 ______ NEW

NC4000 User's Group's Newsletters. Many contributions from Chuck Moore, Rick VanNorman, C.H. Ting and many other users. Hardware enhancements, software and many; utility programs.

803 — MORE ON NC4000, VOLUME 3 $15/16/18 ______ NEW

NC6000/5000 data sheets, quans, new DROP, DEPTH, Eaker's CASE, PICK, ROLL,floating point math packages, new power sources and A/D converters for NC4000. Many other tips.

804 — MORE ON NC4000, VOLUME 4 $15/16/18 ______ NEW

Chuck Moore's Application Notes 1-7, Tiny Modula-2 by Lohr, F83 extensions and other tips from Bill Muench, VanNorman's screen editor. Ting's 32-bit engine design and Fourier transform.

FORTH INTEREST GROUP

P.O. BOX 8231 SAN JOSE, CALIFORNIA 95155 (408)277-0668

Name ______________________
Member Number ______________________
Company ______________________
Address ______________________
City ______________________
State/Prov. ____________ ZIP ____________
Country ______________________
Phone ______________________

OFFICE USE ONLY

By ______	Date ______	Type ______	
Shipped by ______		Date ______	
UPS Wt. ______		Amt. ______	
TNT Wt. ______		Amt. ______	
USPS Wt. ______		Amt. ______	
BO Date ______		By ______	
Wt. ______		Amt. ______	

ITEM #	*TITLE*	*AUTHOR*	*QTY*	*UNIT PRICE*	*TOTAL*
109	*MEMBERSHIP* →				*SEE BELOW*

SUB-TOTAL	
ORDERS OF $50.00 OR MORE RECEIVE A 10% DISCOUNT	
SUB-TOTAL	
CA. RESIDENTS ADD SALES TAX	
HANDLING FEE	*$2.00*
MEMBERSHIP ☐ ***NEW*** ☐ ***RENEWAL $30/42***	
ENCLOSED IS $30/42 FOR 1 FULL YEARS DUES. THIS INCLUDES $24/36 FOR FORTH DIMINSIONS.	

☐ CHECK ENCLOSED (Payable to: Forth Interest Group)

☐ VISA ☐ M/C

Card # ______________________

Expiration Date ______________________

Signature ______________________

($15.00 minimum on all VISA/MC orders.)

PAYMENT MUST ACCOMPANY ALL ORDERS

MAIL ORDERS
Send to:
Forth Interest Group
P.O. Box 8231
San Jose, CA 95155

PHONE ORDERS
Call 408/277-0668 to place credit card orders or for customer service. Hours: Monday-Friday, 9am-5pm PST.

PRICES
All orders must be prepaid. Prices are subject to change without notice. Credit card orders will be sent and billed at current prices. $15 minimum on charge orders. Checks must be in US$, drawn on a US Bank. A $10 charge will be added for returned checks.

POSTAGE & HANDLING
Prices include shipping. A $2.00 handling fee is required with all orders.

SHIPPING TIME
Books in stock are shipped within five days of receipt of the order. Please allow 4-6 weeks for out-of-stock books (delivery in most cases will be much sooner).

SALES TAX
Deliveries to Alameda, Contra Costa, San Mateo, Los Angeles, Santa Cruz and San Francisco Counties, add 6½%. Santa Clara County, add 7%; other California counties, add 6%.

IX-2

fig-FORTH

EXECUTION SECURITY

G.R. JAFFRAY, JR.

One reason Forth runs so fast is that it does not have the built-in checks of other high-level languages. But during development, this means frequent crashes, and the annoying need for hardware reset and reentry of new words into the dictionary.

This can be avoided with a temporary patch to **NEXT**, the Forth thread interpreter. This patch decreases execution speeds, but provides considerable execution security, removing annoying crashes.

The method is to examine CFAs for valid contents: CFA+2 for code words, and one of six values for high-level words (060D = **DOCOL**, 0669 = **DOVAR**, 067B = **DOUSE**, 064F = **DOCON**, 0AC8 = **DODOE**, and 020D = **IDO**). A jump is made to 0103H for any other.

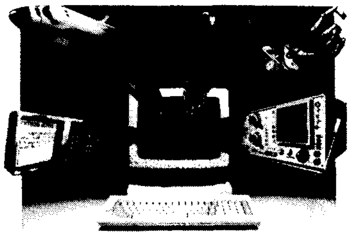
Note: execution time is 25 to 50% longer. Remove patch after debugging.

[Editor's note: It would be interesting to make this table-driven, and to have the table automatically updated by defining words. Slowness shouldn't be too much of a drawback here.]

Patch 8080 fig-FORTH as follows:
Put `JMP XSPTCH` at 014B.

```
XSPTCH: MOV E,M
        INX H
        MOV D,M
        INX H
        MOV A,D
        CMP D
        JNZ XSPTC1
        MOV A,E
        CMP L
        JNZ XSPTC1
        DCX H
        JMP 014EH
XSPTC1: DCX H
        MVI A,06H
        CMP D
        JNZ XSPTC3
        MVI A,0DH
        CMP E
        JZ 014EH
        MVI A,69H
        CMP E
        JZ 014EH
        MVI A,7BH
        CMP E
        JZ 014EH
XSPTC2: JMP 0103H
XSPTC3: MVI A,0AH
        CMP D
        JNZ XSPTC4
        MVI A,0C8H
        CMP E
        JNZ 0103H
        JMP 014EH
XSPTC4: MVI A,02H
        CMP D
        JNZ 0103H
        MVI A,0DH
        CMP E
        JNZ 0103H
        JMP 014EH
```

ASK FORTH ENGINEERING ABOUT REAL TIME THAT'S ON TIME.



Find Out How To Implement Real-Time Systems In:

- Digital Signal Processing
- Manufacturing Process Control
- Machine Vision
- Robotics

...on time and under budget.

For The Answers To Your Questions, Call Our Engineering AnswerLine Today:

(213) 372-8493, Ext. 444.

FORTH, Inc., 111 N. Sepulveda Blvd., Manhattan Beach, CA 90266.

ON TIME. UNDER BUDGET.

FORTH, Inc.

1987 ROCHESTER FORTH CONFERENCE

JERRY SHIFRIN, SYSOP - EAST COAST FORTH BOARD

Thanks to Larry Forsley and company for putting on another fine Forth conference at the University of Rochester. As usual, the meeting had a number of excellent speakers and presentations, with a high level of good technical discussion. I was happy to meet some of the folks we've been typing at for a while: Bob Brown, John Hall, Dennis Ruffer, Mike Ham, Jim Callahan, Bob Davis, and others. Amazing how no one looks the way you imagine they might.

I had a brief chat with Glen Haydon about Loglan (currently being discussed on the ECFB); one interesting note was that Chuck Moore reviewed some of the early Loglan notes, and that there may have been some cross-fertilization between the two languages.

John Hall reported that the FIG GEnie network was still on hold. I passed on the ECFB files (about 30 or so disks) to Dennis Ruffer for use on the FIG GEnie network. It'll be nice to see this get a wider audience.

There were several talks on the massively parallel processors (11,520 16,384 separate processors) at Goddard and Lockheed, both of which are being, at least partly, programmed in Forth. George Nicol of Silicon Composers described their ability to connect multiple, Novix-based coprocessor cards to an IBM PC/AT. So far, they've been able to drive up to eight boards at a time, giving an aggregate capability of 56 MIPS (or MOPS)! In terms of raw compute power, this is easily in the mainframe range. George said that his goal is to get twenty of these things going at once so they'd be in the Crayballpark.

A few new pubs were available — a third edition of Thea Martin's excellent *Bibliography of Forth References*, the *1986 FORML Conference Proceedings*, and an interesting, but overpriced book ($16.95 for 90 pages) called *Forth: The NEXT Step* by Ron Geere. I was tempted to dismiss this book because it's based on fig-FORTH and Forth-79, and a number of its examples use 6502 code definitions.

We are seeing Forth-based applications in several advanced-technology areas.

But it actually has a fair amount to offer: numerous tools on math, time/calendar manipulation, trig, large numbers, sorting, rational numbers, etc. There's some useful stuff here, but $16.95?

We had an interesting panel discussion on Forth standards. The panel consisted of Mahlon Kelly, Larry Forsley, Martin Tracy, and myself, moderated by Jim Basile. There was no general theme; people seemed generally in favor of an ANSI standard, but didn't want anyone tinkering with the language. Glen Haydon said that standards were for people who had nothing better to do. He wanted to see standardization based on common usage and indicated that he was compiling a book on this. Dick Miller simply didn't want any changes, calling himself a conservationist. Other people objected to a self-appointed group deciding on new standards. Bob Brown suggested that Forth look towards Lisp for a model of standardization, due to its many similarities. I apologize if I missed or misrepresented any of the salient points.

Portions of the following are taken directly from the session notes distributed at the conference. Many fascinating presentations are not reviewed here at all, due to space limitations, but you can find the complete proceedings in *The Journal of Forth Application and Research* (Volume 5, Number 1).

Sessions

"The Massively Parallel Processor," John Dorband (NASA Goddard Space Flight Center). John described this two-dimensional array of 16,384 bit-serial processors. It is programmed in Parallel Forth, based on Unified System's Uni-Forth.

"High-Density Parallel Processing: I. The Processor Array and Macro Controller; II. Software and Programming," Nguyen, Raghaven, C. H. Ting, and Truong (Lockheed Palo Alto Research Laboratories). Another huge array of processors — this one has only (!) 11,520 individual processors. Dr. Ting noted that this one obeys the rule "Never trust a computer you can't lift," but he seemed to conclude that meant without chassis and power supply.

"Parallel Processing using the PC-4000 RISC System," George Nicol (Silicon Composers). George described the Novix NC4016-based, add-on boards for the PC/XT/AT, along with their PCX multitasking control and communications system which allows you to interact with each of the processors via separate windows.

"Machine Comprehension as a Control and Planning Tool," Henry Harris (JPL). Henry, always a step ahead of the pack, gave a fascinating, but much too brief, talk on "conceptual dependency," the idea of extracting the deeper, semantic meaning from natural language. He gave just a glimmer of the potential applications of this: translation (of course), data discrimination, bandwidth conservation, and more.

"Mathcalc," David Jagerman (AT&T Bell Laboratories). David put together a fairly extensive math package which provides numerous queuing, statistical, and telephony-oriented computational facilities. This may be available from him without charge (depending on Bell Labs' policy).

"ClusterFORTH in the Factory," Adam Shakleford (FORTH, Inc.). Described FORTH, Inc.'s facility for networking IBM PCs and Z80-based, single-board computers.

"Forth and CAI From Mainframe to Micro: From Coursewriter to Forth," Brooks J. Breeden (Ohio State University). Brooks had one of the best presentations this year, describing his use of Forth-based graphics systems for teaching aspects of landscape architecture.

"Six RePTIL Verbs and the Macintosh," Dr. Israel Urieli (College of Engineering, Ohio University). Noted that, "We have bred a community which genetically accepts Ctrl-Alt-Del as a meaningful statement."

"A Concurrent Architecture for Real-Time Intelligent Process Control," Jack Park (Visiting Scientist, USAF). Jack talked on their PC/AT-M68000 (coprocessor) based multiprocessor expert system.

"Object-Oriented LocalVariables/Data Structures for F83," Robert H. Davis. Bob talked about a number of extensions he made to F83. These files are downloadable from the ECFB.

"LMI Forth for OS/2," Dr. Ray Duncan (Laboratory Microsystems, Inc.). Ray described and demonstrated his Forth system for the long-awaited OS/2 system.

"Toward an Iconic Forth," Gregory Dickerhoof (SARNS-3M). An interesting talk about icons for user interface. Showed several examples of Macintosh applications that use icons for most of the user interface.

"High-Performance Networks," Dr. William Dress (Martin Marietta). Bill gave a fascinating talk about his experiments with programming a bug that learns from its activities.

"The Biological Aspects of Neural Networks," Dr. Iben Browning. Dr. Browning's talk was humorous, literate, and incisive. He discussed the potential for AI, robotics, and consciousness, combined with some scary ideas about how AIDS will impact our society. Dr. Browning was also the after-dinner speaker at the Friday evening banquet, where he expanded on the AIDS threat. A few people walked out on this talk, expressing disagreement or disbelief. In any case, this one won't go away by being ignored.

"Harris Force Processor," David Williams (Harris Semiconductor). Described FORCE (Forth Optimized RISC Computing Environment), a high-speed system suitable for real-time control systems, digital signal processing, etc. It is configurable with numerous "standard cells" (hardware modules) available from Harris. It can be used as a co-processor, stand-alone, or parallel-processing system.

"Chuck Moore's Non-linear Least Squares Fitting Routine using polyFORTH on the Silicon Composers NC4016 Board," Elizabeth Rather (FORTH, Inc.). Typical of Chuck's code, this package fits on 2.5 screens (about 25 lines of code).

"VME-based Language Processor," Tom Harkoway (Xycom). Based on Allen Winfield's MetaForth system. In describing performance, Tom defined MIPS as "Meaningless Information Provided by Salespeople."

Working Groups

Unfortunately there was only one round of concurrent working groups this year and I felt obligated to attend the one on Forth standards. Elizabeth Rather moderated, gave a recap of recent history, and described the ANSI/CBEMA process and Technical Committee membership requirements. I described the bulletin

COMBINE THE RAW POWER OF FORTH WITH THE CONVENIENCE OF CONVENTIONAL LANGUAGES

HS/FORTH

Now you can compile even large programs in the blink of an eye. If you don't need to compile the huge fast programs we handle so well, use the metacompiler to spin off compact ones — as small as a few hundred bytes for simple threaded utilities — as small as 2 kbytes for a full Forth execution core. HS/FORTH is the best base from which to spin off either direct or indirect threaded systems, small or large, or anything else you might invent. This is absolutely the most flexible Forth system available, at any price, with no expensive extras you'll need to buy later. Distribute metacompiled tools, or turnkeyed applications royalty free.

Although HS/FORTH is unmatched for language experimentation and development, remember that we wrote it to be a top notch application development system. Your application will have all of DOS at its disposal, commands, other programs, functions, everything! I/O is easier than in Pascal or Basic, but much more powerful — whether you need parsing, formatting, or random access. Send display output through DOS, BIOS, or direct to video memory. Windows organize both text and graphics display. Math facilities include both software and hardware floating point plus an 18 digit integer (finance) extension and fast arrays for all data types. The hardware floating point covers the full range of trig, hyper and transcendental math including complex. Forth gives you total control of your computer, but only HS/FORTH gives you implemented functionality so you aren't left hanging with "great possibilities" (and lots of work!)

We can't possibly cover everything in this ad, so please call or write and ask for our brochure. We'll also be happy to answer any questions.

HS/FORTH complete system:	$395.
Forth: Text & Reference (500pg)	
Kelly&Spies, Prentice Hall	$ 22.
HS/FORTH: Tutorial & Reference	
Kelly&Callahan (500pg)	$ 59.
HS/FORTH Glossary	$ 10.
DEMO DISK	$ 10.

Visa Mastercard

HARVARD SOFTWORKS

PO BOX 69
SPRINGBORO, OH 45066
(513) 748-0390

board, initially funded by MCI, that will be provided for ANS Forth announcements, proposals, etc. Also discussed was Guy Kelly's plans for Forth extension proposals to be communicated via the new FIG conference on GEnie. A general discussion followed, but again there was no consensus on standards activity, though most people agreed on the need to minimize changes in a new standard.

Vendor Meeting

There were a number of concurrent vendor/user meetings. I attended Laboratory Microsystems' meeting hosted by Ray Duncan and Mike Ham. About 20 people attended. Ray announced they would be shipping PC/Forth Release 3.2 in July. A number of enhancements were suggested. Most of these were acceptable to Ray, and he said they would try to squeeze them into the July release.

News

Look for Ray Duncan to have another bestseller next year when Microsoft Press releases his new book, *Advanced OS/2 Programming*, in January 1988. Let's hope this little gold mine doesn't totally distract him from Forth development.

There were more people using 32-bit Forth implementations this year than previously. The days of the 16-bitters appear numbered, other than for specialized, ROM/SBC applications.

Summary

This year we are seeing real, Forth-based applications coming out of several advanced-technology areas: multi-processors, object-oriented programming, Lisp-like data structures, AI, and the Novix Forth chip. Also notable is Forth usage in some brand new areas: massively parallel processors and neural networks. It's fascinating to see Forth adapt so readily to these new frontiers.

Expert systems seem to have been something of a disappointment, with few people reporting applications or new advances in this area. I wonder whether this is just a lull in activity, or have expert systems fizzled out?

This conference seemed a bit less organized than in previous years, and I was sad to learn that Thea Martin is leaving her position as editor of the *Journal of Forth Application and Research*. Still, it was as enjoyable as ever, with lots of good conversation and opportunities to chat with folks I've previously known only by their typed output. Five days at this conference were worth a great deal in terms of information, education, and conversation.

This review was adapted from the East Coast Forth Board (703-442-8695) with permission from the author/sysop. Jerry Shifrin works for MCI, and is one of the best sysops anywhere, if the tree can be judged by its fruit.

ADVERTISERS INDEX

FOR TRS-80 MODELS 1, 3, 4, 4P
IBM PC/XT, AT&T 6300, ETC.

The MMSFORTH System.
Compare.

- A total software environment: custom drivers for printer, video and keyboard improve speed and flexibility. (New TRS-80 M.4 version, too!)
- Common SYS format gives you a big 395K (195K single-sided) per disk, plus a boot track!
- Common wordset (79-Standard plus MMSFORTH extensions) on all supported computers.
- Common and powerful applications programs available (most with MMSFORTH source code) so you can use them compatibly (with the same data disks) across all supported computers.
- Very fast compile speeds and advanced program development environment.
- A fantastic full-screen Forth Editor: Auto-Find (or -Replace) any word (forward or back), compare or Pairs-Edit any two ranges of blocks, much more.
- Temporary dictionary areas.
- QUANs, VECTs, vectored I/O, and many more of the latest high-performance Forth constructs.
- Manual and demo programs are bigger and better than ever!
- Same thorough support: Users Newsletter, User Groups worldwide, telephone tips. Full consulting services.
- Personal Licensing (one person on one computer) is standard. Corporate Site Licensing and Bulk Distribution Licensing available to professional users.

MMSFORTH

A World of Difference!

The total software environment for IBM PC/XT, TRS-80 Model 1, 3, 4 and close friends.

- Personal License (required):
  MMSFORTH V2.4 System Disk $179.95
  (TRS-80 Model 1 requires lowercase, DDEN, 1 40-track drive.)
- Personal License (additional modules):
  FORTHCOM communications module . . . . $ 49.95
  UTILITIES 49.95
  GAMES 39.95
  EXPERT-2 expert system 69.95
  DATAHANDLER 59.95
  DATAHANDLER-PLUS (PC only, 128K req.) 99.95
  FORTHWRITE word processor 99.95
- Corporate Site License Extensions . . . . from $1,000
- Bulk Distribution . . . from $500/50 units.
- Some recommended Forth books:
  FORTH: A TEXT & REF. (best text!) . . . . $ 18.95
  THINKING FORTH (best on technique) . . . 16.95
  STARTING FORTH (popular text) 19.95

Shipping/handling & tax extra. No returns on software.

Ask your dealer to show you the world of MMSFORTH, or request our free brochure.

MILLER MICROCOMPUTER SERVICES
61 Lake Shore Road, Natick, MA 01760
(617) 653-6136

STARFLIGHT
STAR BRIGHT

AN INTERVIEW WITH TIM LEE

These are the voyages of the video game Starflight. *The programming crew set about creating strange, new worlds, bravely going into greater and more realistic detail than a sci-fi, role-playing simulation — or any other game — had gone before. Michael Ham continues his special series of interviews, with Tim Lee of the imaginative — and persevering — team whose creation continues to break sales records throughout the explored universe.*

MH: You work for Binary Systems, which developed a software product for Electronic Arts. Could you describe the product?

TL: It's a role-playing game in outer space — Star Wars, where you are one of the characters.

MH: It runs on an IBM PC and takes two full disks of program and data, right?

TL: Less 40K. Yes, virtually.

MH: That's only 10K empty per side, so that's still pretty full. I will say that my son has been playing this game non-stop since he got it. He even maintains a log in a little spiral book to keep track of where he is and what he's done. It's quite a game. How many people were working on it when the product came to market?

TL: There were five members of the team. One administrative and logistics person, and one designer and three programmers.

MH: You programmed all this in Forth?

TL: And assembly language.

MH: Forth and assembly language. Now is this the original crew?

TL: Three or four members have come and gone, spending some time on the project.

MH: How long has the project been going on?

There are three, cooperating expert systems in the game.
You can represent a repeatable, fractally generated planet with just a couple of numbers.

TL: Three and one-half years.

MH: When did you come on board?

TL: About six months into it.

MH: It didn't start out quite this big in concept, did it? Did somebody say, "Let's write a program that'll be the biggest game ever released?"

TL: I wasn't there at the time, but it's my understanding from those who were that it was to be a very big-field game. But the idea of fractally generating unique planets that you can return to...

MH: Talk a little bit about fractally generated planets.

TL: It's using the technique developed by Mandelbrot. He takes, essentially, a repeatable random sequence of numbers based on a seed. Using that, you can, in a very compact way, represent a planet with just a couple of numbers, then run it through a function and use that to determine the terrain for the surface of the planet.

MH: So each one is unique but doesn't take up that much room, is that the advantage?

TL: That's the advantage.

MH: David Boulton was one of the early principles.

TL: Right. He really had an impact on shaping the possibility of doing a huge environment. He wrote the first prototype fractal routines.

MH: What does the designer do?

TL: I hesitate to define the roles too exactly, because there is a good deal of design and feedback, give and take.

MH: When I played the game, I encountered these interesting alien races, and with wonderful speech patterns.

TL: That was great. The personality of the game is Greg Johnson. He was the designer. That's what I mean. He designed the scenario; the programmers' role was more to design the game system that you could write the scenario for. You can design subsequent scenarios and not be a technical person; the programmers created a game system that could be expanded by a nonprogrammer designer.

MH: How much did those races and their characteristics evolve as you worked on it? Did they stay pretty much as he first designed them, or did you decide some of them were too evil or too simple-minded?

TL: The races were a collaborative effort — Greg Johnson, Paul Reiche III (who worked on *Archon*), myself, and a number of other people. We got into a sort of roundtable, but the bulk of them were done by Greg Johnson. His playfulness is manifested throughout the entire game.

MH: I don't want to go into how many alien races there are, because it's constant surprise when you are out there and you bump into someone. But did they change? You have the idea for a race. You implement it. Then you interact with them... Did you then decide, no, it's too much this or too many that?

TL: Yes, there was tuning the personality of the race. We had a data record called the race personality record, or something like that, and the designer established and modified parameters.

MH: The dialog is great. Is that also Greg's?

TL: That is also Greg. And also, again, Paul Reiche and a number of other people.

MH: Was this the first product Electronic Arts signed?

TL: No, but it was one of the early ones.

MH: It was on the first weekly status report, I recall hearing that.

TL: Yes, it's the only Electronic Arts product that has been tracked, to date, on all the different status report forms they've ever had.

MH: So it was the longest in development and, of course, the most ambitious in scope. It was released when?

TL: End of August 1986.

MH: And sold out instantly, as I recall. And still going strong.

TL: As I understand, yes.

MH: I see notices about it on the boards, by the way. People are saying, "I can't believe it."

Fifth 2.5

- ***Emphasis on Programming in the Large:***
  Tree Structured Scoping of Dictionary
  Direct Editing of Dictionary Structure with Dictionary Editor
  Text Editor allows Screens of any Size
  Large Memory Model, 32-bit Stack, Arithmetic
- ***Tight Binding of Source and Code:***
  Modifed Modules are Compiled upon leaving the Text Editor
  New Definition Completely Replaces the Old Definition in Dictionary
  Old Definition is Returned to Free Memory
  Implements Compile by Demand
  Avoids Re-Loading Source Files
- ***Online Help Facility:***
  F1 Key Provides Context Sensitive Help (on Errors, in Editors)
  Provides Quick Reference on Primitives
  Apropos Help from Text Editor on both Primitives and User Defined Modules
- ***Turnkey Application Generator:***
  Produces a Stand Alone EXE File
  Strips Unused Primitives, Kernel Routines
  Invoked with a Single Keystroke
  Vectored Error Handling
- ***Complete Debugging Tools:***
  Source Level Tracing, Breakpoints
  Inspect or Modify Variables during Trace
  Shell to the Interpreter During Trace

TL: That's great!

MH: What aspect of the game gave you the most problems, what was the most difficult thing to accomplish?

TL: The — enclosure of the design.

MH: You set a monster in motion that didn't want to close down?

TL: Yes, that was really tough. And communicating among the programmers was also a challenge.

MH: Did you work in one place, or did each one work at home?

TL: We did a little bit of both. Bob Gonsalves, one of the other programmers, and I worked at the Binary Systems location, and Bob was getting down a couple of days a week — he was the other programmer. So that's the whole crew. Myself, Bob, Alec, Greg, and Rod on the administrative end. Rod McConnell was really the entrepreneur, the one who brought the team together.

MH: In communicating among them, how did you handle the word explosion? Large Forth projects tend to generate a lot of words. Did you maintain any sort of dictionary, or did you brief each other, did you read each other's code? How did you know what words other people had developed?

TL: We segmented things hierarchically, so one person was responsible for primitives, stack operatives, and system primitives.. And everybody else used those primitives. So, of course, when you work with primitives, we'd document them (that was me, in this case) and build a glossary of graphics words. It would take a week or two of the other guys using this thing, referencing the document, for them to fully understand how the words worked. Then they'd say, "I've got a great idea for a primitive, let's put it in...." Then it would require looking at how many bytes were involved here and there. One of the more challenging things was squashing the amount of source code we have on those disks. We have over four megabytes of source.

MH: Really! Screen files or text files?

TL: Screen files, so there is a lot of white space in many places. The text is pretty dense, and with the aliens' speech is a lot of text.

MH: (grinning) And of course, the source code is contains lots of comments, right?

TL: Well, our styles vary.

MH: With really good code, you don't need comments. (laughs)

TL: (grins) That's my latest feeling. Write long definitions, and the comments and the code are the same thing. In fact, the product chronicles the development of

---

Fifth 2.5

- ***Provides Complete Programming Tools:***
  - ***Primitives for Dynamic Memory Support***
  - ***Produces Native Code - Very Fast***
  - ***Complete Access to MS-DOS Files***
  - ***8087 Floating Point Support***
  - ***Provides Range Checking***
  - ***Graphics***
- ***Includes Fifth Source Files:***
  - ***Inline 8086, 8087 Assembler***
  - ***Forth 83 to Fifth Convertor***
  - ***Infix Expression Compiler***
- ***A Shareware Version (Fifth 2.0) is Available***
  - ***Lacks Some Features of Fifth 2.5***
  - ***Runs Most Fifth 2.5 Programs***
  - ***May Be Freely Distributed***

For IBM PC's with 128K, DOS 2.0 or better.

Professional Version:	***$150.00***
Shareware Version, Disk and Manual:	***$ 40.00***
Shareware Version, Disk:	***$ 10.00***

System Source Code Available

CLICK Software
P.O. Box 10162
College Station, TX 77840
(409)-696-5432

MS-DOS is a registered trademark of Microsoft Corp.
IBM is a registered trademark of International Business Machines Corp.

our programming styles. The earlier code was really terrible, and you can see the conceptual hurdles we leaped as we went on.

MH: I noticed that, too, in my programming. When I look at the early stuff, I can't believe I wrote it. Does your program come in in overlays?

TL: Yes, there are about sixty overlays.

MH: I heard in some discussions that there is a bit of artificial intelligence in it — adaptability to the player.

TL: Yes, there are, I think, three separate expert systems in the game. That was another thing I was involved with. We wrote an expert system that allowed the game designer, who was not a programmer, to design the rules for the behavior of the aliens and communication module, how the ships behaved, how the life forms on the planets work — all of those were cooperating expert systems. He wrote the rules for them, and he didn't have to know Forth or any programming.

MH: And that freed the programmers from much tedious work.

TL: That's right, and it allowed him to tune the thing without needing us in the loop. It was great.

MH: What are the expert systems?

TL: There's one for handling the aliens on the surface of a planet, one for the aliens in space — that's the ship movement rules — and one for communications, handling the personality of the talking character.

MH: You lived with that project for a long time. Knowing what you know now, what would you do differently?

TL: I'd have to think about it a little bit to give you a real good answer.

MH: We're satisfied with a mediocre answer at this time. You can always tune it.

TL: I think I would have recognized, based on the knowledge I've acquired, that we were designing a game, an entertainment product, and not a model of the universe. We spent a lot of time trying to model things that didn't add to the play of the game. I mean *Starflight*, believe it or not, is a vastly cut-down subset of the original.

MH: You went for verisimilitude with a vengeance — you wanted the whole thing. In fact, you ended up with how many stars and planets?

TL: There are 810 planets and 200 star systems.

MH: And you can visit any of them in any sequence. The player is free to move around.

TL: That's right.

MH: So you put a lot of time and effort into making ever-finer details of reality. I would think that, having a lot of people working on it, you'd get to challenging each other to get further and further into it.

TL: Yes. The whole thing acquired a personality. The morale of the team went up and down in cycles, based on a number of factors, the least of which was paying the rent.

MH: Yes, three and one-half years is a long development cycle, especially when you are paid on milestones. Just out of curiosity, are all of you science fiction fans?

TL: Yes.

MH: So you were into this as an art form already. Was your concept influenced during the course of this development, as various science fiction movies came out? This was after the first *Star Wars*?

TL: Oh, yes — much later.

MH: Was it influenced by anything you saw or read as you went along?

TL: A great deal. We have a rotating planet, which was why I originally hired in. The rotating planet figured prominently in *Star Wars*, and we had to have it. You see that view of the planet turning as you're watching it.

MH: So this is how you got in, because they wanted the planet to turn.

TL: Yeah, and I could do that.

MH: You were the graphics guru...

TL: Graphics, virtual memory management, cache system, and object-handling data base, which was very significant in allowing the program modules to talk to each other.

MH: What is an object-handling data base?

TL: When I say "object handling," it's really a fancy name for a list manager, a pre-structured list manager. And I wrote a language for manipulating objects on this tree.

MH: And the object in this case would be a planet description, for instance?

TL: Planet description, alien ship... it was open-ended, in the sense that you could take an object anywhere in the universe, and then we defined a record description for that object, and we'd write code that would look at the fields within that object.

MH: What were the other programmers' specialties?

TL: Bob Gonsalves spent a great deal of his time generating the life forms and the ecosystems for the life-bearing planets. He spent many, many months on that. He also designed the interactional planet site. The planet site is pretty much Bob Gonsalves.

MH: The planet site where you are seeking out minerals and encountering aliens.

TL: Dave Boulton's prototypes created a

3-D terrain. I adapted that to map it onto a sphere, making it a planet. The planet builder, which places minerals, life forms, and other planetary features, was mostly Bob's work.

MH: The planet descriptions are extremely detailed regarding the eco-system, the atmosphere, and the mass; depending on whether you've trained your science officer. If you haven't trained your science officer, the basic description of the planet is, "Don't know." But if you've got a well-trained science officer, you have an amazing amount of detail.

TL: It's hard to talk about the game in terms that someone else would understand. I took the fractal system that generated the planet and displayed the graphics. But the life-form responses — where they move around a lot, the terrain vehicle interface, all of the functions on the terrain vehicle before the player goes into outer space, that's Bob. Alec Kersco singlehandedly wrote the star port module. Everything that happens in star port, he designed code for it, he wrote the code for that. Greg Johnson designed the interface. Actually, he was fundamentally responsible for the design of it. Again, good ideas came from all quarters.

MH: The star port is shopping, getting lists of things, the training, and so on. The player moves a small figure around the six stores, going through the little safari shop.

TL: The other thing Alec was responsible for was the module for communicating with aliens. I helped in the design of that, he actually had it built by another programmer who left the project, then he tore it down and rebuilt it in three weeks with the addition of the expert system.

MH: If, God forbid, all the code was lost and you redid it again from scratch, would it be much more compact?

TL: No question about it.

MH: You really learned a lot?

TL: Oh, yes.

MH: You've never really given serious thought to throwing it out and totally rewriting it...

TL: I have.

MH: Oh, you have! The perfectionist in the Forth programmer arises yet again. If one were to play this on the ideal system, what would you recommend — an AT with a composite color monitor? That's the ideal?

TL: Yes. It has a caching system that takes up about all of available RAM, so you don't even need to have a hard disk. If you have a lot of memory, it will take advantage of it.

MH: The composite monitor is for the better color. You've written an amazing system, and I'm thinking of all the techniques you developed. Do you see a way to apply them in a business application, to store and retrieve data, graphical information? You've got a way of handling data that is interesting and fast. Will you use these tools in your later applications?

TL: Yes. I don't think I will get into the business realm, but I'm certainly going to use the tools that we developed.

MH: Do you think you'll do more games, or are you interested in doing more games after this one? They're only three and a half years each, is the way I see it... in ten years you can do three games.

TL: I do have a game that I'd like to do.

MH: Do each of you have a game in mind?

TL: Yeah.

MH: What was your first encounter with Forth?

TL: My first encounter with Forth was as a contractor on a project for tax

Forth Interest Group Presents

Ninth Annual

FORTH NATIONAL CONVENTION

November 13-14, 1987

Red Lion Inn
2050 Gateway Place
San Jose, California 95110

FIG's 10th Anniversary Celebration
EXHIBITS
CONFERENCE PROGRAM

THE EVOLUTION OF FORTH
Past - Present - Future

The founders, writers, producers, and evaluators of Forth look at:

Forth in development
Forth at work
Forth in the future

Conference & Exhibit Hours

Fri. Nov.13	12 noon - 6 pm
Sat. Nov. 14	9 am - 5 pm

Hotel Reservations

Telephone Red Lion Inn, San Jose
800 547-8010 or 408 279-0600
Request special Forth Interest Group Rates.

Convention Registration

Preregistration	$20
Registration at the door	$25
Banquet Saturday 7 p.m. (with keynote speaker)	$35

___Yes, I will attend the convention.

Preregistrations @ $20 $_____
Banquet Tickets @ $35 $_____
Total check to FIG (US funds) $_____

Name ____________________
Address ____________________
Company ____________________
City ____________________
State ________ ZIP ____________
Country ____________________
Phone (____) ______________

Return to: Forth Interest Group
P. O. Box 8231
San Jose, CA 95155

Information Phone: (408) 277-0668

preparation. It was to be written in BASIC, because at that time the only thing that was really common was BASIC. So it was going to be in BASIC, but for the work we had to do, we decided the best thing short of going to assembly language would be Forth. We used C.H. Ting's Forth interpreter written in BASIC and buried that in the product. It was fast enough for going through the form and doing calculations.

MH: Did you get into computers through a major?

TL: In high school, I read a book about computers and they had an example of a tic-tac-toe playing computer made with matchboxes. That was the very first step. But then it was about six years before I could get onto a real computer, and that was at the University of Illinois at Champaign-Urbana, the Plato system. I was in high school at the time, and a friend of mine showed it to me. I stole access time, I begged, borrowed and stole keys, I wrote a fake sign-on thing that system access, and we could then create all of our own passwords from then on in. At that time, the morality of my access didn't occur to me. Having to do it over again, I would do it differently; but nevertheless, I wrote my first games when I was about, let's see, seventeen.

MH: In BASIC?

TL: TUTOR. That's the Plato system.

MH: And what was TUTOR like?

TL: It's very loose, I don't remember much about it. It seemed like a mongrel language. BASIC-like commands. I don't think it had line numbers.

MH: When you went to college, were you going to be in computers right from the start?

TL: No, I was going to be a dentist, because my mom wanted me to be a dentist. So I went for two semesters. Well, I went for two semesters and I couldn't take the garbage. At the same time, I was working part-time at Texas Instruments in the programmable calculator division, in their software exchange. So that's where my real education took place, and when I quit going to school I just went full-time at Texas Instruments.

MH: What were you doing for TI?

TL: I was evaluating programs written for the SR56, the SR52, and the TI59 programmable calculators. People would submit programs, and we'd send them free copies of other programs in return. The programs would then be published in booklets.

MH: When was that?

TL: 1977 and '78. I was there while they were designing the TI 99/4A.

MH: That could have been a great computer.

Visit the MACH 2 Product Support RoundTable™ on GEnie™ !!

MACH 2

Multi-tasking FORTH 83 Development System

MACH 2 FOR THE MACINTOSH™ 99.95

features full support of the Macintosh toolbox, support of the Macintalk speech drivers, printing and floating point, easy I/O redirection and creates double-clickable, multi-segment Macintosh applications. Includes RMaker,disassembler, debugger, Motorola-format (infix) 68000 assembler and 500 pg manual.

MACH 2 FOR THE OS-9 OPERATING SYSTEM™ 495.00

provides position-independent and re-entrant execution and full support of all OS-9 system calls. Creates stand-alone OS-9 applications. Link FORTH to C and vice-versa. Includes debugger,disassembler, Motorola-format (infix) 68000 assembler, and 400 page manual.

MACH 2 FOR INDUSTRIAL BOARDS 495.00

is 680X0 compatible, provides 68881 floating point support, and produces position-independent, relocatable, ROM-able code (no target compilation required). Includes disassembler, Motorola-format (infix) 68000 assembler, and 350 pg. manual.

PALO ALTO SHIPPING COMPANY
P.O. Box 7430
Menlo Park, California 94026
415 / 854-7994 • 800 / 44FORTH

VISA/MC accepted. CA residents include 6.5% sales tax.
Include shipping/handling with all orders: US $6; Canada $8; Europe $25; Asia $30
RoundTable and GEnie are registered trademarks of the General Electric Information Services Company.

TL: The designers knew they were screwing it up; they didn't have any choice. They were dealing with other specifications that they hadn't developed, and they said, "You know, we could make these teeny little changes, it would be great! But we can't." They had to follow this huge set of specifications.

MH: Bureaucracy rears its ugly head. How did you get to California?

TL: I came out on vacation and I just decided to stay. This is the greatest place.

MH: Was it in California that you hit Forth?

TL: Yes. Prior to that time I had never heard of it, and I'd done a lot of reading and programming, because it was a hobby. But I never heard of Forth. That would have been 1979. That was seven years since it hit the public scene.

MH: You liked it, apparently.

TL: Loved it. After seeing how it worked under BASIC, I felt a little bit ludicrous putting in these SWAPs and DROPs and shoo-bops. And then I tried to think about how I would write an interpreter, if I was going to do it; and, hey, there was no simple way. Then I got to understanding what colon definitions were, because these were just strings of things that [would go in the] interpreter and I could rule the world with this!

MH: Yeah, this is dangerous...

TL: Never look back!

MH: Was *Starflight* your first major Forth project?

TL: I'd done a Forth video game for Datasoft, called *Genesis*, and it was never shipped in the IBM format because they didn't want to ship a Forth product. It ran as fast as the assembly language versions of the Apple and the Commodore versions, and yet they didn't want to ship it.

MH: Religious differences, I take it.

TL: Well, that and they wanted to make sure it ran on the PCjr, which at that time was hot property. I decided to leave at that time and they didn't have anybody else, so it never got out.

MH: So *Starflight* was your first, giant project of original work—

TL: —in Forth. My first giant project was as Vice-President of Datax Corporation, a financial-planning computer-services company that did policy analysis for life insurance companies. I wrote more lines of programming, in a language called PICK-BASIC on Quantel's computers, than I care to think about.

MH: Is that the small computer the agents carried into the field?

TL: No, this was a minicomputer. I think Unisys bought them. I had fifteen employees at the high point, and I managed to spend half a million dollars of someone else's money. So I've been an entrepreneurial type, and like to live on the riskier side of things than being in the corporate environment. After I got out of Texas Instruments, I decided corporations came with too much politics. If anything needed to get done, and you went by the book, it never happened. If you knew people and exchanged pull and favors, you managed to get things done. That was the way things happened.

MH: How long do you think it takes to learn Forth? I thought I knew Forth fairly early in this project, but now I realize I didn't really. What does it take to learn Forth so that your code is reasonably fluent and sound and easy to follow? It takes more than just reading *Starting Forth.*

TL: My own philosophy is that you need to write your own Forth.

MH: From the ground up?

TL: When you get to the point where you understand how to write your own Forth from the ground up, you are ready to write good Forth code.

MH: A demanding requirement. There is a book that takes one through that: *ThreadedInterpretativeLanguages*.It'sfor the Z80. I'd hate to do write my own Forth, though. I like to buy it out of the box.

We've covered quite a bit here. Suppose a reader decides, "That's a great idea. I'll write a computer game and make a lot of money because it's so popular, and I'll go out and be rich and famous, too." What would you say to such a person?

TL: I would say, if that's what you want to do, do it. But I tell you, the road is rocky. I don't mean in any way to dissuade someone from doing what they like to do. For me, it's been worth it, but it's been tough. As far as making lots of money, if we sell one hundred thousand units of Starflight, which is really a lot for a game — I think ultimately we will sell that many — then for three and a half years, I will have been making $6.50 an hour.

MH: *Starflight*, by the way, because it is so exceptional a game, may have a longer shelf life than most. You're saying, though, that if someone wants to write a game, it's interesting and they can learn lots, but they shouldn't quit their job.

TL: No. Keep your day job, that's important.

MH: If you were going to write a game, what computer would you write it for?

TL: The IBM PC.

MH: Really!

TL: Yes.

MH: Would you do a graphics game again, or would you make it text only?

TL: I think graphics are so much more appealing, as evidenced by [the fact that] all the really hot products that stay hot products, have graphical stimulators. If you can combine that with depth, then so much the better.

MH: But why the PC?

TL: There's just a lot of them out there, and it's a decent machine to develop on.

MH: The PC has enough power.

TL: It's getting cheap, too.

MH: Pretty soon, if you buy over fifty dollars worth of groceries, you get one free at your supermarket. If you are talking to someone who knows nothing about graphics, and he or she says, "Okay I want to do graphics. How do I do graphics?" Is there a book?

TL: Yes, there are several really good books. One by Newman and Sproull that's called *The Principles of Interactive Computer Graphics*, and another is Foley and Van Dam's, *Fundamentals of Interactive Computer Graphics.*

MH: Tim, thank you very much.

TL: You're welcome.

Michael Ham is a professional Forth programmer who writes extensively and works in large-scale data processing. One of Tim Lee's Starflight *associates wanted us to be sure that, while giving credit all around, Tim accepts his own fair share. We hear that he not only takes a logical approach to problem solving, but applies his own inner spirit of adventure to programming.*

Congratulations to the whole team: Only days ago, Starflight *received Electronic Arts' awards for "Best Creation of New Worlds" and "Best Entertainment Product." Maybe the Forth really* is *with them!*

(Letters-Continued from page 8.)

Bacon's Screens

```
SCR # 18
( Forgetful LOAD  )
   HEX
VARIABLE BUF
: SAYBLOCK    ( u --   )
   BASE @ DECIMAL  HERE A8 + BUF !
   SWAP US>D <#  BL HOLD  # # #  57 HOLD  #>
   BUF @ SWAP CMOVE  BASE !  ;
: LOAD    ( u --  )  DUP SAYBLOCK
    BLK @  0 BLK !  TIB @  BUF @ TIB !  >IN @  0 >IN !
   -FIND  IF  DROP DROP  0 >IN !  FORGET  THEN
   0 >IN !  CREATE  >IN !  TIB !  BLK !  LOAD  ;
: -->    (  --  )  BLK @ 1+
   STATE @  0= IF  DUP SAYBLOCK  0 BLK !
   TIB @  BUF @  TIB !  0 >IN !  CREATE  TIB !  THEN
   0 >IN !  BLK !  ;  IMMEDIATE
   DECIMAL
```

Screen 17 is my improvement to the modification, to return an offset into the buffer, rather than the address of the matched string. This added five bytes to the code size: a **PUSH** to save the buffer address on the stack, a **POP** and a **SUB** in the event of a successful exit, and an additional **POP** in the case of an unsuccessful search (to clean up the stack before returning).

Even with my improvements, this implementation falls short of the full functionality provided by Laxen and Perry, since it is not sensitive/neutral to the case of the pattern or object string (depending on the state of the variable **CAPS**). Admittedly, Brother Zimmerly was designing for use in some specific, but unspecified, application of his own, where this might not be a shortcoming. However, I wish to replace the **SEARCH** provided in the F83 system, so it had better behave exactly as the original. I have included the code for such an assembler version of **SEARCH**.

This code compiles to a whopping 169 bytes — more than twice the size of the code in screen 17. I have made one concession to size by moving the conversion of characters into upper case to a labeled subroutine. In trying to arrive at a satisfactory tradeoff between size and speed, I have left the decision about converting a character with the main **SEARCH** code, only making the call if **CAPS** is on. This decision, along with a **PUSH** and a **POP** of the BX register, which I have used as a working register for this operation, could be moved to the subroutine to save additional bytes of dictionary space, or the whole routine could be embedded into **SEARCH** at the appropriate places, for speed at the expense of size.

While I have offered some critical observations of Brother Zimmerly's code, I wish to thank him for his effort, and *Forth Dimensions* for publishing it. Without this to pique my interest, I wouldn't have done this work. Being a neophyte programmer, I have learned a lot about F83 in the process, and am pleased with the results.

I enjoy the magazine and wish it came monthly.

Sincerely,
Robert Lee Hoffpauer
Richardson, Texas

Mods Quad Divides

Dear Mr. Ouverson:

The enclosed code modifies that of Robert L. Smith (*FD* VIII/6) for a quad-by-double divide in several respects. First, it is in a form suitable for the LMI assembler from which many Forth assemblers are derived. Second, it does the entire procedure in registers, resulting in about a 30% speed-up in the code. Third, it adds error checking by executing the divide-overflow interrupt where appropriate. Fourth, it preserves the BP register (along with **SP** and **SI**), which many 8088 Forths use as the return pointer.

LMI's 32-bit PC/Forth+ has a similar procedure that is bulkier, a shade slower, and proprietary.

Sincerely,
Michael Barr
Mount Royal, Quebec

```
ASSEMBLER LABEL >UP   ASCII a # AL CMP >=  \ Char to uppercase.
     IF ASCII z # AL CMP <= IF 20 # AL SUB THEN THEN RET  FORTH

CODE  SEARCH  (S  sadr slen badr blen --- offset flag  )
   CLD                   \ Set direction of search.
   CX POP   BX POP       \ Get CXbuf-len & BXbuf-adr.
   DX POP   DI POP       \ Get DXstr-len & DIstr-adr.
   DX CX SUB   BX CX ADD \ CXlast-adr (last possible match.)
   SI PUSH     BX PUSH   \ Save Forth IP & BXbuf-adr.
   BX SI XCHG            \ BXbuf-adr to SIbuf-ptr for LODS.
   0 [DI] AL MOV         \ 1st char of string.
   CAPS #) BX MOV   0 # BX CMP   0<> \ Test up/low case.   (JE)
   IF >UP #) CALL THEN   \ Convert char to uppercase if CAPS on.
   AH AL XCHG            \ Swap ah&al.
   BEGIN                 \ MAIN LOOP.
     AL LODS             \ Char from buffer, incr SIbuf-ptr.
     CAPS #) BX MOV   0 # BX CMP   0<> \ Test up/low flag. (JE)
     IF >UP #) CALL THEN              \ If on, convert char.
     AH AL CMP 0=        \ Do they match?                (JNE)
     IF                  \ If equal,
       SI PUSH           \ save current SIbuf-ptr,
       1 # BX MOV        \ Set BXstr-index to 2nd char,
       BEGIN             \ and check rest of string.
         0 [DI+BX] AL MOV \ Char at DIstr-adr + BXstr-index.
         BX PUSH                       \
         CAPS #) BX MOV   0 # BX CMP   0<> \ Test case.   (JE)
         IF >UP #) CALL THEN           \ If on, convert.
         AH AL XCHG                    \ Swap ah&al.
         AL LODS          \ Char from buffer & incr SIbuf-ptr.
         CAPS #) BX MOV   0 # BX CMP   0<> \ Test case.   (JE)
         IF >UP #) CALL THEN           \ If on, convert.
         BX POP                        \
         AH AL CMP   0= \ Do they match?                 (JNE)
       WHILE             \ While matching,
         BX INC          \ increment BXstr-index,
       REPEAT            \ and loop.
       SI POP            \ Restore SIbuf-ptr.
       DX BX CMP   >=    \ Is BXstr-index >= DXstr-len?    (JL)
       IF                \ If so return w/FOUND.
         BX POP          \ Original buffer address.
         BX SI SUB       \ Calculate offset of found string.
         SI DEC          \
         DX POP          \ Restore original Forth IP.
         DX SI XCHG      \ IP to SI, offset to DX.
         -1 # AX MOV     \ True flag signals FOUND!
         2PUSH           \
       ELSE              \ Reload AH w/first char of string.
         0 [DI] AL MOV   \ 1st char of string.
         CAPS #) BX MOV   0 # BX CMP   0<> \ Test case.   (JE)
         IF >UP #) CALL THEN           \ If on, convert.
         AH AL XCHG                    \ Swap ah&al.
       THEN              \
     THEN                \
     CX SI CMP  U>       \ Is SIbuf-ptr > CXlast-adr?     (JBE)
   UNTIL                 \ If so return w/NOT-FOUND, else loop.
   DX POP                \ Use buf-adr for meaningless value.
   SI POP                \ Restore original Forth IP.
   AX AX XOR             \ False flag signals NOT-FOUND!
   2PUSH                 \
 END-CODE                \
    Code size: 169 bytes (including headers for >UP and SEARCH)
```

(Continued)

MCA

FIG-FORTH for the Compaq,
IBM-PC, and compatibles. $35

Operates under DOS 2.0 or later, uses standard DOS files.

Full-screen editor uses 16 x 64 format.

Editor Help screen can be called up using a single keystroke.

Source included for the editor and other utilities.

Save capability allows storing Forth with all currently defined words onto disk as a .COM file.

Definitions are provided to allow beginners to use *Starting Forth* as an introductory text.

Source code is available as an option, add $20.

Async Line Monitor
Use Compaq to capture, display, search, print, and save async data at 75-19.2k baud. Menu driven with extensive Help. Requires two async ports. $300

A Metacompiler on a
host PC, produces a PROM for a target 6303/6803 Includes source for 6303 FIG-Forth. Application code can be Metacompiled with Forth to produce a target application PROM $280

FIG-Forth in a 2764 PROM
for the 6303 as produced by the above Metacompiler. Includes a 6 screen RAM-Disk for stand-alone operation. $45

An all CMOS processor
board utilizing the 6303. Size: 3.93 x 6.75 inches. Uses 11-25 volts at 12ma, plus current required for options. $190 ~~$240~~ - $280

Up to 24kb memory: 2 kb to 16kb RAM, 8k PROM contains Forth. Battery backup of RAM with off board battery.

Serial port and up to 40 pins of parallel I/O.

Processor buss available at optional header to allow expanded capability via user provided interface board.

Micro Computer Applications Ltd
8 Newfield Lane
Newtown, CT 06470
203-426-6164

Foreign orders add $5 shipping and handling. Connecticut residents add sales tax.

PORTABLE POWER WITH MasterFORTH

Whether you program on the **Macintosh**, the **IBM PC**, an **Apple II** series, a **CP/M** system, or the **Commodore 64**, your program will run unchanged on all the rest.

If you write for yourself, MasterFORTH will protect your investment. If you write for others, it will expand your marketplace.

Forth is interactive – you have immediate feedback as you program, every step of the way. Forth is fast, too, and you can use its built-in assembler to make it even faster. MasterFORTH's relocatable utilities and headerless code let you pack a lot more program into your memory. The resident debugger lets you decompile, breakpoint and trace your way through most programming problems. A string package, file interface and full screen editor are all standard features. And the optional target compiler lets you optimize your application for virtually any programming environment.

The package exactly matches *Mastering Forth* (Brady, 1984) and meets all provisions of the Forth-83 Standard.

MasterFORTH standard package	**$125**
(Commodore 64 with graphics)	**$100**
Extensions	
Floating Point	**$60**
Graphics (selected systems)	**$60**
Module relocator (with utility sources)	**$60**
TAGS (Target Applic. Generation System) – MasterFORTH, target compiler and relocator	**$495**
Publications & Application Models	
Printed source listings (each)	**$35**
Forth-83 International Standard	**$15**
Model Library, Volumes 1-3 (each)	**$40**

(213) 821-4340

MICROMOTION

8726 S. Sepulveda Bl., #A171
Los Angeles, CA 90045

Hoffpauer's Screens, continued.

```
Scr # 15        B:TOOLS.BLK
( Zimmerly's search from Forth Dimensions vol VIII # 4 p 5 )
.LOADING HEX
  \
ASSEMBLER LABEL (FIND1)  DX SI MOV BX DX MOV BP POP 2PUSH FORTH
  \
CODE SEARCH (S sadr slen badr blen --- adr tflg | junk fflg  )
  CLD CX POP DI POP BX POP DX POP BP PUSH DX SI XCHG
  CS AX MOV AX ES MOV 0 [SI] AL MOV
  HERE BYTE REP SCAS 0=
  IF CX PUSH SI PUSH DI PUSH DI DEC BX CX MOV BYTE REPZ CMPS 0=
     IF BX POP AX POP AX POP BX DEC -1 # AX MOV (FIND1) #) JMP
     THEN DI POP SI POP CX POP
  ELSE AX AX XOR (FIND1) #) JMP THEN #! JMP END-CODE
.LOADED DEC \S
  \ Code size: 87 bytes including headers for SEARCH & (FIND1)
  \


Scr # 16        B:TOOLS.BLK
( Modified Zimmerly search )
.LOADING HEX
  \
CODE SEARCH (S sadr slen badr blen --- adr tflg | junk fflg  )
  CS AX MOV   AX ES MOV   CLD
  CX POP      DI POP      BX POP      DX POP
  DX SI XCHG  0 [SI] AL MOV
  HERE  BYTE REP SCAS    0=
  IF    CX PUSH   BX CX MOV    SI PUSH    DI PUSH    DI DEC
        BYTE REPZ CMPS    0=
        IF    DX SI MOV   DX POP   DX DEC   4 # SP ADD
              -1 # AX MOV       2PUSH        THEN
        DI POP   SI POP   CX POP   ROT #) JMP
  ELSE  DX SI MOV  AX AX XOR   2PUSH   THEN   END-CODE
.LOADED DEC \S
  \ Code size: 67 bytes including a header for SEARCH.


Scr # 17        B:TOOLS.BLK
( Improved Zimmerly search )
.LOADING HEX
  \
CODE SEARCH (S sadr slen badr blen --- offset tflg | junk fflg )
  CS AX MOV   AX ES MOV   CLD
  CX POP      DI POP      BX POP      DX POP     DI PUSH
  DX SI XCHG  0 [SI] AL MOV
  HERE  BYTE REP SCAS    0=
  IF    CX PUSH   BX CX MOV    SI PUSH    DI PUSH    DI DEC
        BYTE REPZ CMPS    0=
        IF    DX SI MOV   DX POP   DX DEC   4 # SP ADD   BX POP
              BX DX SUB   -1 # AX MOV       2PUSH        THEN
        DI POP   SI POP   CX POP   ROT #) JMP
  ELSE  DX SI MOV  DX POP  AX AX XOR   2PUSH   THEN   END-CODE
.LOADED DEC \S
  \ Code size: 72 bytes including a header for SEARCH.
```

```
Screen # 61
( Quad-division                          MB     13:51 05/12/87 )
ASM86 CODE QUAD/MOD
        CX POP
        DX POP
        AX POP
        BX POP   CX, AX CMP 6$ JA 7$ JNE DX, BX CMP 7$ JNA
    6$: DI POP
        SI PUSH
        BP PUSH
        BP, SP MOV
        SI, 4 [BP] MOV
        BP, CX MOV
        CX, # 32 MOV
        CLC
    1$: SI, 1 RCL
        DI, 1 RCL                          -->


Screen # 62
( Quad-division, cont                          14:02 05/12/87 )
        BX, 1 RCL
        AX, 1 RCL
        3$ JNC
    2$: BX, DX SUB
        AX, BP SBB
        STC
        1$ LOOP
        5$ JMP
    3$: AX, BP CMP
        4$ JC
        2$ JNZ
        BX, DX CMP
        2$ JNC
    4$: CLC
-->


Screen # 63
( Quad-division, end                           14:02 05/12/87 )
        1$ LOOP
    5$: SI, 1 RCL
        DI, 1 RCL
        CX, SI MOV
        BP POP
        SI POP
        DX POP
        BX PUSH
        AX PUSH
        CX PUSH
        DI PUSH
        NEXT,       7$: 0 INT
        END-CODE
```

FORTH SOURCE™

WISC CPU/16

The stack-oriented "Writeable Instruction Set Computer" (WISC) is a new way of harmonizing the hardware and the application program with the opcode's semantic content. Vastly improved throughput is the result.

Assembled and tested WISC for IBM PC/AT/XT	$1500
Wirewrap Kit WISC for IBM PC/AT/XT	$ 900
WISC CPU/16 manual	$ 50

MVP-FORTH

Stable - Transportable - Public Domain - Tools

You need two primary features in a software development package... a stable operating system and the ability to move programs easily and quickly to a variety of computers. MVP-FORTH gives you both these features and many extras.

MVP Books - A Series

☐ **Vol. 1,** *All about FORTH.* Glossary	$25
☐ **Vol. 2,** *MVP-FORTH Source Code.*	$20
☐ **Vol. 3,** *Floating Point and Math*	$25
☐ **Vol. 4,** *Expert System*	$15
☐ **Vol. 5,** *File Management System*	$25
☐ **Vol. 6,** *Expert Tutorial*	$15
☐ **Vol. 7,** *FORTH GUIDE*	$20
☐ **Vol. 8,** *MVP-FORTH PADS*	$50
☐ **Vol. 9,** *Work/Kalc Manual*	$30

MVP-FORTH Software - A transportable FORTH

☐ **MVP-FORTH Programmer's Kit** including disk, documentation. Volumes 1, 2 & 7 of MVP Series, FORTH Applications, and Starting FORTH, IBM, Apple, Amiga, CP/M, MS-DOS, PDP-11 and others. Specify. $175

☐ **MVP-FORTH Enhancement Package** for IBM Programmer's Kit. Includes full screen editor & MS-DOS file interface. $110

☐ **MVP-FORTH Floating Point and Math** ☐ IBM, ☐ Apple, or ☐ CP/M, 8". $75

☐ **MVP-LIBFORTH** for IBM. Four disks of enhancements. $25

☐ **MVP-FORTH Screen editor** for IBM. $15

☐ **MVP-FORTH Graphics Extension** for ☐ IBM or ☐ Apple $80

☐ **MVP-FORTH PADS (Professional Application Development System)** An integrated system for customizing your FORTH programs and applications. PADS is a true professional development system. Specify Computer: ☐ IBM ☐ Apple $500

☐ **MVP-FORTH MS-DOS** file interface $80

☐ **MVP-FORTH Floating Point Math** $100

☐ **MVP-FORTH Graphics Extension** $80

☐ **MVP-FORTH EXPERT-2 System** for learning and developing knowledge based programs. Specify ☐ Apple, ☐ IBM, or ☐ CP/M 8". $100

Order Numbers:

800-321-4103

(In California) **415-961-4103**

FREE CATALOG

MOUNTAIN VIEW PRESS

PO BOX X
Mountain View, CA 94040

CANDIDATES' STATEMENTS

NOMINEES TO THE FIG BOARD OF DIRECTORS

Forth Interest Group members in good standing will elect members of its Board of Directors at the Forth National Convention on November 13-14, 1987. The Board will expand from five to seven members at that time. Robert Reiling and Martin Tracy will continue serving, and John D. Hall is running for reelection.

Kim Harris and Thea Martin are leaving the Board, after donating countless hours to the benefit of the entire Forth community. Kim is one of the five original founders of the Forth Interest Group, and has played an important role ever since then in the shaping of organizational policy and activities, and the Forth language. He will direct the program again at this year's FORML conference.

The following individuals have been proposed by the Nominating Committee as candidates for the five vacant positions on the Board of Directors. Nomination of additional individuals is by petition, requires the signatures of 25 members in good standing of the Forth Interest Group, and must be received by September 19, 1987. A voting mechanism will be provided at the convention.

Nominees were asked to submit statements of their candidacy in 250 or fewer words:

Wil Baden

My first computer experience was the MANIAC at Princeton in the 1940's, but I did not become a programmer until 1960. In the meantime I worked as a translator, private detective, editor, and other interesting but uneconomical jobs. Since then I have worked or consulted for Collins Radio, Marshall Communications, CDC, Corregated Computing Techniques, ARAMCO, Logicon, IBM, General Automation, Honeywell, Burroughs, HP, and others. At present I am "senior tool designer" at Doelz Networks, Inc., and am implementing a compiler for LUCREZIA, a programming language named after Lucretia Borgia, to enable programmers to write poisonous spaghetti code in a high-level language. Outside interests include being emcee-director of a stag show with over a thousand performances coast to coast; and lay reader, chalice-bearer, and religious instructor in the Episcopal Church. I have a wife, two daughters, two sons (one a computer hardware expert, the other a software expert, but both excellent Forth programmers), and a small assortment of dogs, cats, and personal computers. I sing second tenor.

John D. Hall

I appreciate being nominated once again as a director of the Forth Interest Group. I am willing to serve and I am looking forward to continued active participation.

The first five years of the Forth Interest Group showed rapid growth and many starts in several directions. The current three years have set us on a firm financial and organizational foundation. When I accepted the directorship, I set three goals to work toward. First was the consolidation and stabilization of FIG Chapters. Second was the growth of income to stabilize the foundation for future growth of the organization. Third was the consolidation and coordination of all Forth organizations into the effort of promoting Forth.

The first two of these goals have now been fully realized and, in accepting the nomination and if elected, I will continue with my third goal and I will set three other goals. First, we should broaden our reach to include professional Forth programmers and companies with products based on Forth. We can and should expect an extraordinary effort on their part in supporting the Forth Interest Group. Second, we must expand and support all educational efforts which promote Forth, whether through FORML, or programs by our members and chapters, or at universities. Third, we must provide additional communication programs to support these first two goals.

The Forth Interest Group will continue to stand as the centerpiece in the promotion of Forth.

Dennis Ruffer

Not many of you have heard of me, so I should explain my background. Fifteen years ago, I took a BASIC programming course and, liking it so much, I switched career directions to Computer Science. Four years later, I graduated from Western Michigan University with a major in Mathematics and a minor in Computer Science (only because they did not offer a Major in CS). I started my career in Data Processing, with RPG. Mastering that, nine years ago I was offered a job in engineering. The rest is the history of the Smart Scope, an automobile-diagnostics tool produced by the Test Products Division of the Allen Group. We now produce four products using Forth, with eight programmers and a patented diagnostic method I developed.

But why should I direct the Forth Interest Group? My goal is to increase the acceptability of Forth as a viable computer language. Forth suffers from a bad image, an image that it is unsuitable for large applications, that the principles of Computer Science do not work with Forth. I, for one, do not agree with this image, but the image is not completely unjustified. Forth is not much different than many other languages; we still need to use the tools of Software Engineering, but I have not seen much effort in that direction. Our founding fathers (of Forth) have created a very fine language, but software is more than just a language. The software life cycle includes requirements, design, testing, and maintenance, in addition to the implementation of the code. I would like to help this industry move beyond the implementation details, and on to solving the management concerns. Hopefully, I will at least make an impression.

Robert L. Smith

I have been actively involved in the Forth community for 10 years, beginning with my membership in the fig-FORTH Implementation Team. I have attended a majority of business meetings of FIG, and have been a member of the board of our local FIG Chapter. I have been the Secretary of the Forth Standards Team since 1982. I have used Forth professionally for the majority of time since learning it. I am the author of the "Standards Corner," which appeared in *Forth Dimensions*, and have also published a number of other Forth papers.

If elected to the Board, I will strive to represent the various parts of the Forth community. My particular interest is in the needs of the professional Forth programmer. As a member of the Board, I would be eager to hear suggestions for improving the Forth Interest Group.

Teri Sutton

My name is Terri Sutton, and I am candidate for the FIG Board of Directors. I have been a Forth programmer for over six years. I am currently programming real-time firmware, using Forth assembler. I have just finished a one-year term as Treasurer on the Board of Directors of the Silicon Valley FIG Chapter. I traveled to Taiwan and China last fall to talk about the history of FIG. These experiences have led me to the conclusion that FIG isn't sure what its role should be. As a board member, I will work to determine clear goals for the organization and how to implement them. For example, I want FIG to be able to support its members better by offering workshops and seminars. I would like to look into improving the employment service now offered. Forth isn't getting the credit it deserves, and companies that could profit from using Forth are choosing not to use it. I think FIG should try to change this, possibly by educating programmers and/or companies. I believe that, as a Board member, I can improve FIG's future.

Go FORTH™ ***The*** ProDOS Forth Language implementation for the Apple Computer //e, //c, //gs and ///

FORTH is more than just a high level language that combines many of the features of other computer languages. It is a development environment and a method of approaching problem solving. FORTH is a 'grass roots' language, developed and enhanced in the real world by working programmers who needed a language that they could USE. Many of the concepts of FORTH are several years ahead of other languages of today. It is a language as interactive as Applesoft Basic, yet, unlike Applesoft, you don't have to pay the price in slow execution speed. Programs written totally in FORTH are usually faster than programs written in C or Pascal and a heck of a lot smaller. Best of all, FORTH has a large library of public domain programs.

Go FORTH is the new FORTH language implementation for the Apple® //e, //c, //gs (//e emulation mode, full //gs version late Fall) and the Apple® ///. It is 100% ProDOS® and SOS ® supported. ***Go FORTH*** code is intercompatable with all ***Go FORTH*** supported machines machines. ***Go FORTH*** is for the hobbyist, the systems developer, the applications writer, anyone who wants to learn and use the powerful FORTH language.

Go FORTH comes with its manual and an assortment of utilities in its SCREEN file. Many other utilities and suppor t systems will be available soon. For beginners, we highly recommend the Starting Forth manual, and we recommend the ***Go FORTH*** Toolkit series for everyone!

ONLY $69.95 Complete, order #5807

Go FORTH Toolkit #1 (Applsoft-like commands/utilities):	$49.95, order #5809
Starting Forth by Leo Brodie (The training manual for Forth):	$21.95, order #5706

Add $1.00 Shipping and handling per item.

24 HOUR VISA / MASTERCARD ORDER LINES

California Only: (800) 541-0900. Outside California: (800) 334-3030. Outside U.S.A. : (619) 941-5441

PAIR SOFTWARE (916) 485-6525

3201 Murchison Way, Carmichael, California 95608

Apple //e, //c, //gs and ///, ProDOS and SOS are registered trademarks of Apple Computer, Inc. No affiliation with Pair Software

FIG CHAPTERS

U.S.A.

• ALABAMA
Huntsville FIG Chapter
Tom Konantz (205) 881-6483

• ALASKA
Kodiak Area Chapter
Horace Simmons (907) 486-5049

• ARIZONA
Phoenix Chapter
4th Thurs., 7:30 p.m.
Dennis L. Wilson (602) 956-7578
Tucson Chapter
2nd & 4th Sun., 2 p.m.
Flexible Hybrid Systems
2030 E. Broadway #206
John C. Mead (602) 323-9763

• ARKANSAS
Central Arkansas Chapter
Little Rock
2nd Sat., 2 p.m. &
4th Wed., 7 p.m.
Jungkind Photo, 12th & Main
Gary Smith (501) 227-7817

• CALIFORNIA
Los Angeles Chapter
4th Sat., 10 a.m.
Hawthorne Public Library
12700 S. Grevillea Ave.
Phillip Wasson (213) 649-1428
Monterey/Salinas Chapter
Bud Devins (408) 633-3253
Orange County Chapter
4th Wed., 7 p.m.
Fullerton Savings
Huntington Beach
Noshir Jesung (714) 842-3032
Sacramento Chapter
4th Wed., 7 p.m.
1798-59th St., Room A
Tom Ghormley (916) 444-7775
San Diego Chapter
Thursdays, 12 noon
Guy M. Kelly (619) 454-1307
Silicon Valley Chapter
4th Sat., 10 a.m.
H-P, Cupertino
George Shaw (415) 276-5953
Stockton Chapter
Doug Dillon (209) 931-2448

• COLORADO
Denver Chapter
1st Mon., 7 p.m.
Clifford King (303) 693-3413

• CONNECTICUT
Central Connecticut Chapter
Charles Krajewski (203) 344-9996

• FLORIDA
Orlando Chapter
Every other Wed., 8 p.m.
Herman B. Gibson (305) 855-4790
Southeast Florida Chapter
Coconut Grove area
John Forsberg (305) 252-0108
Tampa Bay Chapter
1st Wed., 7:30 p.m.
Terry McNay (813) 725-1245

• GEORGIA
Atlanta Chapter
3rd Tues.,6:30 p.m
Western Sizzlen, Doraville
Nick Hennenfent (404) 393-3010

• ILLINOIS
Cache Forth Chapter
Oak Park
Clyde W. Phillips, Jr.
(312) 386-3147
Central Illinois Chapter
Urbana
Sidney Bowhill (217) 333-4150
Rockwell Chicago Chapter
Gerard Kusiolek (312) 885-8092

• INDIANA
Central Indiana Chapter
3rd Sat., 10 a.m.
John Oglesby (317) 353-3929
Fort Wayne Chapter
2nd Tues., 7 p.m.
I/P Univ. Campus, B71 Neff Hall
Blair MacDermid (219) 749-2042

• IOWA
Iowa City Chapter
4th Tues.
Engineering Bldg., Rm. 2128
University of Iowa
Robert Benedict (319) 337-7853
Central Iowa FIG Chapter
1st Tues., 7:30 p.m.
Iowa State Univ., 214 Comp. Sci.
Rodrick Eldridge (515) 294-5659
Fairfield FIG Chapter
4th day, 8:15 p.m.
Gurdy Leete (515) 472-7077

• KANSAS
Wichita Chapter (FIGPAC)
3rd Wed., 7 p.m.
Wilbur E. Walker Co.,
532 Market
Arne Flones (316) 267-8852

• MASSACHUSETTS
Boston Chapter
3rd Wed., 7 p.m.
Honeywell
300 Concord, Billerica
Gary Chanson (617) 527-7206

• MICHIGAN
Detroit/Ann Arbor area
4th Thurs.
Tom Chrapkiewicz (313) 322-7862

• MINNESOTA
MNFIG Chapter
Minneapolis
Even Month, 1st Mon., 7:30 p.m.
Odd Month, 1st Sat., 9:30 a.m.
Vincent Hall, Univ. of MN
Fred Olson (612) 588-9532

• MISSOURI
Kansas City Chapter
4th Tues., 7 p.m.
Midwest Research Institute
MAG Conference Center
Linus Orth (913) 236-9189
St. Louis Chapter
1st Tues., 7 p.m.
Thornhill Branch Library
Contact Robert Washam
91 Weis Dr.
Ellisville, MO 63011

• NEW JERSEY
New Jersey Chapter
Rutgers Univ., Piscataway
Nicholas Lordi (201) 338-9363

• NEW MEXICO
Albuquerque Chapter
1st Thurs., 7:30 p.m.
Physics & Astronomy Bldg.
Univ. of New Mexico
Jon Bryan (505) 298-3292

• NEW YORK
FIG, New York
2nd Wed., 7:45 p.m.
Manhattan
Ron Martinez (212) 866-1157
Rochester Chapter
4th Sat., 1 p.m.
Monroe Comm. College
Bldg. 7, Rm. 102
Frank Lanzafame (716) 235-0168
Syracuse Chapter
3rd Wed., 7 p.m.
Henry J. Fay (315) 446-4600

• NORTH CAROLINA
Raleigh Chapter
Frank Bridges (919) 552-1357

• OHIO
Akron Chapter
3rd Mon., 7 p.m.
McDowell Library
Thomas Franks (216) 336-3167
Athens Chapter
Isreal Urieli (614) 594-3731
Cleveland Chapter
4th Tues., 7 p.m.
Chagrin Falls Library
Gary Bergstrom (216) 247-2492
Dayton Chapter
2nd Tues. & 4th Wed., 6:30 p.m.
CFC. 11 W. Monument Ave., #612
Gary Ganger (513) 849-1483

• OKLAHOMA
Central Oklahoma Chapter
3rd Wed., 7:30 p.m.
Health Tech. Bldg., OSU Tech.
Contact Larry Somers
2410 N.W. 49th
Oklahoma City, OK 73112

• OREGON
Greater Oregon Chapter
Beaverton

2nd Sat., 1 p.m.
Tektronix Industrial Park,
Bldg. 50
Tom Almy (503) 692-2811
Willamette Valley Chapter
4th Tues., 7 p.m.
Linn-Benton Comm. College
Pann McCuaig (503) 752-5113

• **PENNSYLVANIA**
Philadelphia Chapter
4th Sat., 10 a.m.
Drexel University, Stratton Hall
Melanie Hoag (215) 895-2628

• **TENNESSEE**
East Tennessee Chapter
Oak Ridge
2nd Tues., 7:30 p.m.
Sci. Appl. Int'l. Corp., 8th Fl.
800 Oak Ridge Turnpike,
Richard Secrist (615) 483-7242

• **TEXAS**
Austin Chapter
Contact Matt Lawrence
P.O. Box 180409
Austin, TX 78718
Dallas/Ft. Worth
Metroplex Chapter
4th Thurs., 7 p.m.
Chuck Durrett (214) 245-1064
Houston Chapter
1st Mon., 7 p.m.
Univ. of St. Thomas
Russel Harris (713) 461-1618
Periman Basin Chapter
Odessa
Carl Bryson (915) 337-8994

• **UTAH**
North Orem FIG Chapter
Contact Ron Tanner
748 N. 1340 W.
Orem, UT 84057

• **VERMONT**
Vermont Chapter
Vergennes
3rd Mon., 7:30 p.m.
Vergennes Union High School
Rm. 210, Monkton Rd.
Don VanSyckel (802) 388-6698

• **VIRGINIA**
First Forth of Hampton Roads
William Edmonds (804) 898-4099
Potomac Chapter
Arlington
2nd Tues., 7 p.m.
Lee Center
Lee Highway at Lexington St.
Joel Shprentz (703) 860-9260
Richmond Forth Group
2nd Wed., 7 p.m.
154 Business School
Univ. of Richmond
Donald A. Full (804) 739-3623

• **WISCONSIN**
Lake Superior FIG Chapter
2nd Fri., 7:30 p.m.
Main 195, UW-Superior
Allen Anway (715) 394-8360
MAD Apple Chapter
Contact Bill Horton
502 Atlas Ave.
Madison, WI 53714
Milwaukee Area Chapter
Donald Kimes (414) 377-0708

INTERNATIONAL

• **AUSTRALIA**
Melbourne Chapter
1st Fri., 8 p.m.
Contact Lance Collins
65 Martin Road
Glen Iris, Victoria 3146
03/29-2600
Sydney Chapter
2nd Fri., 7 p.m.
John Goodsell Bldg., Rm. LG19
Univ. of New South Wales
Contact Peter Tregeagle
10 Binda Rd., Yowie Bay
02/524-7490

• **BELGIUM**
Belgium Chapter
4th Wed., 20:00h
Contact Luk Van Loock
Lariksdreff 20
2120 Schoten
03/658-6343
Southern Belgium Chapter
Contact Jean-Marc Bertinchamps
Rue N. Monnom, 2
B-6290 Nalinnes
071/213858

• **CANADA**
Northern Alberta Chapter
4th Sat., 1 p.m.
N. Alta. Inst. of Tech.
Tony Van Muyden (403) 962-2203
Nova Scotia Chapter
Halifax
Howard Harawitz (902) 477-3665
Southern Ontario Chapter
Quarterly, 1st Sat., 2 p.m.
Genl. Sci. Bldg., Rm. 212
McMaster University
Dr. N. Solntseff (416) 525-9140
ext. 3
Toronto Chapter
Contact John Clark Smith
P.O. Box 230, Station H
Toronto, ON M4C 5J2
Vancouver Chapter
Don Vanderweele (604) 941-4073

• **COLOMBIA**
Colombia Chapter
Contact Luis Javier Parra B.
Aptdo. Aereo 100394
Bogota 214-0345

• **DENMARK**
Forth Interesse Gruppe Denmark
Copenhagen
Erik Oestergaard, 1-520494

• **ENGLAND**
Forth Interest Group- U.K.
London
1st Thurs., 7 p.m.
Polytechnic of South Bank
Rm. 408
Borough Rd.
Contact D.J. Neale
58 Woodland Way
Morden, Surry SM4 4DS

• **FRANCE**
French Language Chapter
Contact Jean-Daniel Dodin
77 Rue du Cagire
31100 Toulouse
(16-61)44.03.06
FIG des Alpes Chapter
Annely
Georges Seibel, 50 57 0280

• **GERMANY**
Hamburg FIG Chapter
4th Sat., 1500h
Contact Horst-Gunter Lynsche
Common Interface Alpha
Schanzenstrasse 27
2000 Hamburg 6

• **HOLLAND**
Holland Chapter
Contact Adriaan van Roosmalen
Heusden Houtsestraat 134
4817 We Breda
31 76 713104

• **IRELAND**
Irish Chapter
Contact Hugh Dobbs
Newton School
Waterford
051/75757 or 051/74124

• **ITALY**
FIG Italia
Contact Marco Tausel
Via Gerolamo Forni 48
20161 Milano
02/435249

• **JAPAN**
Japan Chapter
Contact Toshi Inoue
Dept. of Mineral Dev. Eng.
University of Tokyo
7-3-1 Hongo, Bunkyo 113
812-2111 ext. 7073

• **NORWAY**
Bergen Chapter
Kjell Birger Faeraas, 47-518-7784

• **REPUBLIC OF CHINA (R.O.C.)**
Contact Ching-Tang Tzeng
P.O. Box 28
Lung-Tan, Taiwan 325

• **SWEDEN**
Swedish Chapter
Hans Lindstrom, 46-31-166794

• **SWITZERLAND**
Swiss Chapter
Contact Max Hugelshofer
ERNI & Co., Elektro-Industrie
Stationsstrasse
8306 Bruttisellen
01/833-3333

SPECIAL GROUPS

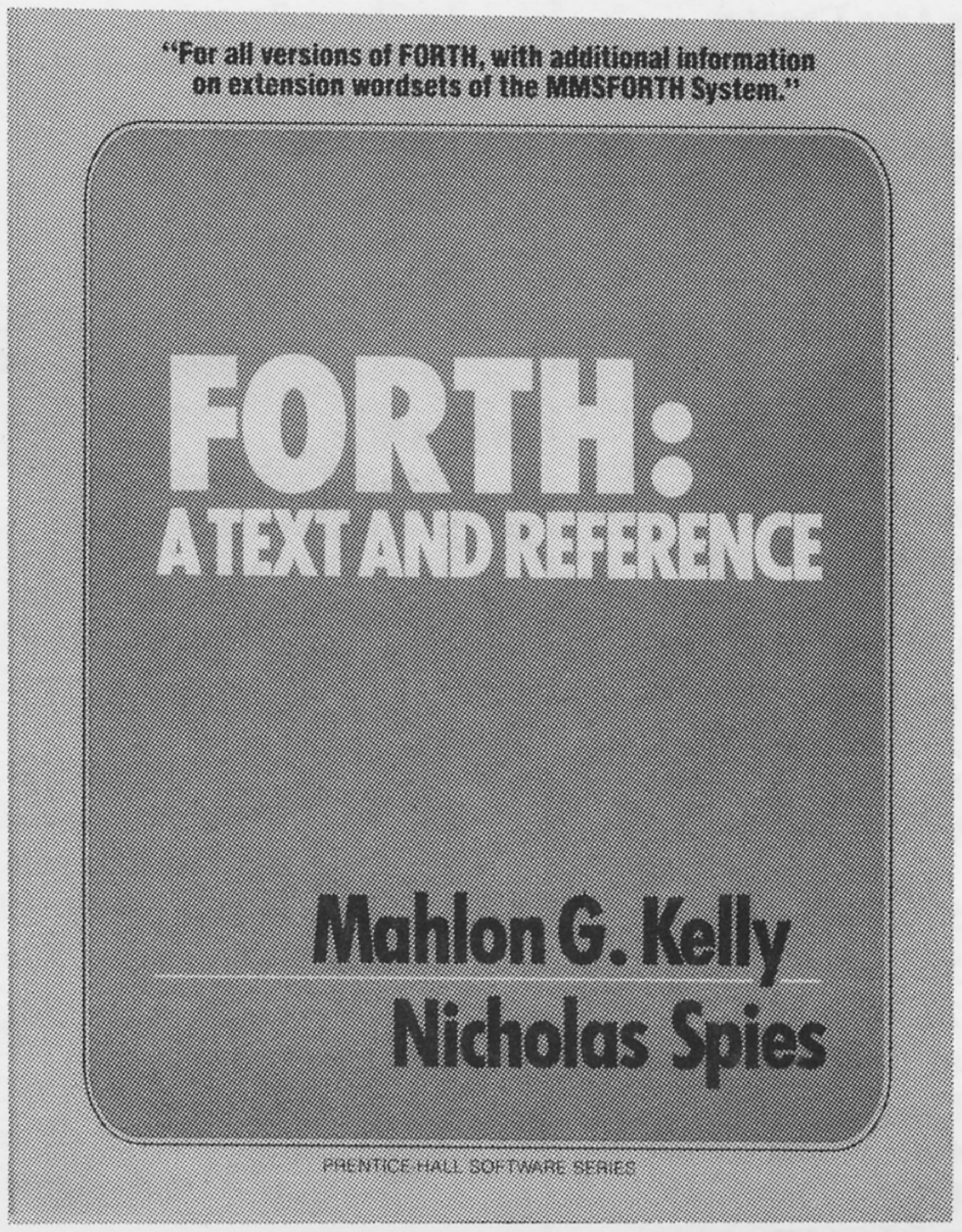
• **Apple Corps Forth Users Chapter**
1st & 3rd Tues., 7:30 p.m.
1515 Sloat Boulevard, #2
San Francisco, CA
Dudley Ackerman
(415) 626-6295

• **Baton Rouge Atari Chapter**
Chris Zielewski (504) 292-1910

• **FIGGRAPH**
Howard Pearlmutter
(408) 425-8700

• **NC4000 Users Group**
John Carpenter (415) 960-1256

NOW AVAILABLE


"For all versions of FORTH, with additional information on extension wordsets of the MMSFORTH System."
FORTH: A TEXT AND REFERENCE
Mahlon G. Kelly
Nicholas Spies
PRENTICE-HALL SOFTWARE SERIES

FORTH INTEREST GROUP fig

$22 EACH

FORTH INTEREST GROUP fig

FROM THE FORTH INTEREST GROUP

Forth Interest Group
P.O.Box 8231
San Jose, CA 95155