MATCHPOINT

THE VISIBLE FORTH

GRIDPLOT

FIRST ANS FORTH MEETING
IT'S ALL HERE:
SPEED, TOOLS, FLEXIBILITY
THE PC-RISC SYSTEM

■ SPEED
  • NC4016 RISC Engine
  • 4 MHz operation runs at 5 MIPS
  • 5 MHz operation runs at 6 MIPS
  • 6 MHz operation runs at 7 MIPS
  • Up to 40 MIPS with multiple PC4000s

■ TOOLS
  SCForth2
  • Multitasker switches tasks in 26 cycles
  • Compiler optimizes over 150 phrases
  • Loads from block or ASCII text files

  PCX
  • PC/PC4000 interface software with windowing
  • On line help screens and full featured editor
  • Concurrent PC operation

  SC-C
  • K&R standard implementation for NC4016
  • Full 64K word address space
  • In-line assembly code

  SCMMacro
  • Full featured native code assembler
  • Includes linker, editor and image generator

■ FLEXIBILITY
  • Coprocessor board for PC, XT, or AT
  • General purpose development system
  • Ideal for control or image processing
  • OEM product inclusion

  PC4000 4MHz . . . . . . . . . . . . . . $1,295  SCForth2 . . . . . . . . . . . . . . . . . $295
  PC4000 5MHz . . . . . . . . . . . . . . $1,495  SC-C . . . . . . . . . . . . . . . . . . . . $595
  PC4000 6MHz . . . . . . . . . . . . . . $1,695  SCMMacro . . . . . . . . . . . . . . . . . $295

SILICON COMPOSERS, 210 California Avenue, Palo Alto, CA 94306  (415) 322-8763
DRILL AND PRACTICE • BY RICHARD H. TURPIN
9

The ease of number conversion in Forth is at the heart of this simple drill-and-practice program from a course in digital design. Beginning Forth programmers will find examples of variables, printing to the screen, receiving keyboard responses, and comparison operations. Typical of many Forth applications is the ease with which variations can be generated — try it!

MATCHPOINT • BY J. BROOKS BREEDEN
12

One strength of computer-assisted instruction (CAI) is repetition. But a quiz should not ask questions in the same order every time, or students may learn the test rather than the subject matter. This paper describes a simple method for thoroughly “scrambling” a multiple-choice, matching CAI quiz, using an example based on highway superelevation.

THE VISIBLE FORTH • BY RICH FRANZEN
18

This decompiler displays source code along with the addresses for each element. Use it to explore Forth’s dictionary structure, and to write quick patches. The author compares its power to both a sledgehammer and a jeweler’s screwdriver.

ANS FORTH MEETING NOTES • BY JERRY SHIFRIN
27

Last August, the ANS Forth Technical Committee (X3/14) met for the first time. Many leaders of the Forth community were present, including Elizabeth Rather as acting chairperson. The effort was described as documenting common practice, not as using this standard as an instrument for change.

GRIDPLOT • BY GENE THOMAS
30

Forth allows users of the TI 99/4A to operate in high resolution without assembly language. But the transcendental functions used by graphics compete — disastrously — with the bit-map color table for work space! A sine table will work, but Bumgarner’s method of trig derivation is better. Add major changes to the graphics commands, and GRIDPLOT is born.

EDITORIAL
4

LETTERS
5

ADVERTISERS INDEX
35

FIG CHAPTERS
38
Two articles this month will help you to dabble with Forth-based CAI (computer-assisted instruction): Breeden’s “Matchpoint” is a bite-sized portion of a full-featured CAI engine. It deals with an interesting problem for CAI designers. Those with little experience in Forth, or with less interest in CAI, may prefer experimenting with the simpler “Drill and Practice” code by Turpin. A number of enhancements suggest themselves that would be good programming exercises for beginners.

If you are one of those who sits on the bench, listening to Forth hackers and understanding some of their theory, but with little personal experience spelunking in Forth’s murkier depths, “The Visible Forth” may be a good way to take your first plunge. It’s a decompiler, which is good for starters, but the more you use it, the more it reveals about what makes your Forth tick (or ‘`). Pretty soon, you’ll be hacking around and messing up control structures with the best of ’em. And probably jotting down ideas for another better NEXT (or TO, or LOOP, or CASE…).

Martin Tracy called to make sure our readers know that the ANS Forth Technical Committee is accepting technical proposals. Mail proposals to Martin (acting secretary) at FORTH, Inc., 111 N. Sepulveda Blvd., Manhattan Beach, California 90266. And look for Martin’s new Forth column in Dr. Dobb’s Journal of Software Tools’ October issue.

An upcoming event of interest: this year’s Forth convention will mark a ten-year anniversary, an excellent time not only to see where Forth has come from, but to discuss and learn more about where it is headed. Convention organizers have scheduled presentations by key people from Forth’s history, and by those who are now shaping the language’s next decade.

But don’t wait until then to join the FIG-sponsored “RoundTable” (e.g., special-interest area) on GENie, General Electric’s information service. It features an extensive library of Forth software, real-time chats with other on-line Forth programmers, a FIG Chapters section, and an area for public messages. Like similar networks, you can call this one from most parts of the U.S. via a local phone call, and GE plans to expand its international service.

Many thanks to sysops Dennis Ruffer, Scott Squires, Gary Smith, and John Hall, for their work getting the Forth RoundTable ready for use. Let them know what you think. Many of the files are from the East Coast Forth Board, graciously donated by Jerry Shifrin as seed material. (Before you dive unprepared into the GEnie discussion of Forth standards, check Jerry’s notes in this issue from the first meeting of the ANS Forth committee.)

For details of the November Forth Convention, and of GEnie’s sign-up offer for FIG members, refer to the FIG ads elsewhere in this issue. And don’t forget FORML — see you there!

—Marlin Ouverson
Editor
Fractal’s Dimension

Dear Marlin,

First, let me congratulate you on the new look of Forth Dimensions. The cover is particularly striking, and the style is quite effective.

I particularly enjoyed the article on fractal landscapes, which I have awaited eagerly since I saw it demonstrated at the last FORML conference. The waves were a nice touch.

Unfortunately, I could not find Figures One-a to One-d referred to in the article.

Most importantly, I would like to point out that the third paragraph of the article is somewhat misleading. A fractal is an object with a fixed dimension. As the article stated, most coastlines have lengths which increase as we measure them with finer and finer precision. However, the dimension of the coastline remains fixed.

Perhaps all will become clearer if I define a version of fractional dimension known as “capacity.” Cover the fractal \( F \) with a number of balls (or disks), each of diameter \( d \). Count the number of balls, then let \( d \) get smaller, and count the minimum number of balls of the new diameter it takes to cover \( F \); continue in this manner. If \( N(d) \) is the number of balls of diameter \( d \) it takes to cover \( F \), then the capacity, or dimension, of the fractal may be defined as the following limit:

\[
\lim_{d \to 0} \frac{\log N(d)}{-\log d}
\]

As a simple example, consider the Cantor set \( C \), which is a fractal created by removing the middle third from the unit interval \([0,1]\), then removing the middle third (of length \(1/9\)) from each of the remaining intervals \([0,1/3]\) and \([2/3,1]\), and so on. Now it takes two balls (or intervals) of diameter \(1/3\) to cover \( C \), while it takes four balls of diameter \(1/9\) to cover \( C \). In general, \(2^n\) balls of diameter \(1/3^n\) will cover the set. Thus, the dimension of \( C \) is \(\log 2/\log 3\).

There are other notions of dimension, such as the classical Hausdorff Dimension, in which one covers the set with convex sets rather than balls, but the above definition is one of the simplest.

On another topic, I converted Mr. Koopman’s excellent source code to F83. I used my own F83 assembler version of Bresenham’s line-drawing algorithm for the PC, and added the ability to save and restore pictures to and from disk (also in assembler), as well as a menu with a few other options.

Thanks,
Mark Smiley
Department of Mathematics
Auburn University at Montgomery
Montgomery, Alabama 36193-0401

Brand Name Dropping

Dear Mr. Ouverson,

I am composing a list of commercially available software that is (or was at one time) written using Forth. I request anyone knowing of such products to contact me via the East Coast Forth BBS (703-442-8695) or at my address below. Also, the source of the information should be included (e.g., “Rapidfile,” per Ashton-Tate’s 3/87 Technotes). Following is my list to date:

VP-Planner
Zoomracks I & II
VALDOCS

Fractal Landscapes Figures.
Run DOS Files in F83

Dear Editor,

I would like to contribute a method to run F83 programs written in standard PC-DOS text files.

Like many users of Laxen and Perry's F83 Forth, I immediately looked for a way to write programs with my favorite text editor (PC-Write). Conversion of ASCII text files to Forth blocks, and vice versa, is not built into F83. Some very nice programs to do so have appeared in Forth Dimensions (e.g., "Screenless Forth," VIII/5).

It is possible, however, to run programs written in ASCII text files without any addition to F83. To do so, simply redirect the input stream to come from the text file.

There are two catches: In the text file, an end-of-line usually takes the form of a carriage-return/line-feed. F83 expects only a carriage-return. If you use PC-Write as an editor, then it is a simple matter to remove all line-feeds from the file. The second problem to overcome is inherent in the use of I/O redirection. At the end of the text file, F83 will wait for more input from the file. This can cause a "hanging" of the system. You can solve this by including a SAVE-SYSTEM and a BYE at the end of the text file.

Below is a simple example of how the method works. The text file MYPROG.TXT contains a Forth program followed by the SAVE-SYSTEM and BYE. The resulting Forth system, MYFORTH.COM, contains an executable version of the program:

```
MYPROG.TXT, the ASCII text of the Forth program:
: binary (-) 22 base ! ;
SAVE-SYSTEM my£orth.com
BYE
```

DOS commands to get an executable version of MYPROG:

```
F 83 < myprog.txt
myforth
```

A good, full-screen editor for the Forth environment is still preferable, but this is a handy workaround. I hope others will benefit from it.

Yours sincerely,
Richard de Rozario
Unit 8.6
Waverton
Sydney, Australia

F83 Gets Gregorian

Finally, Matt Wilson from Yagoona, Australia, sent his F83 version of Allen Anway's Gregorian date routine (FD IX/1), "...together with enough to get a calendar listing." Further elaborations, anyone?

—Ed.
Wilson’s F83 calendar screens:

```
1 0 \ perpetual calendar primitives FDXlp34
2 GREEDAY (5 year month day -- ud)
3  ROT ( MM DD YY/YYYY )
4  DUP 100 \ IF 1900 + THEN
5  ROT ( DD YYYY MM ) \ special, prev year
6  DUP 3 \ IF 12 + SWAP
7  1- \ SWAP THEN
8  2.3059 100 \ ( month calc. )
9  ROT \ YYYYUDD)
10 32 \ YYYYU)
11  ROT \ UD YYYY)
12  DUP \ UD N YYYY)
13  DUP4001 ROT!* \ YYYYUDD)
14  36525 \ Ult
15 100 \ lU/lOD
16  ( UD D rod UD
17  ROT \ D+ D+ ;
18
2 0 \ perpetual calendar primitives FDXlp34
1 WEKDAY ( S UD -- mod ) ( gregorian-day -- 0-6)
2 \ 0= Sunday, 1=Monday, 2=Tuesday, etc. )
3  3 0 D+ 7 MU/MOD 2DROP ;
4
5 NTH-SUNDAY ( S year month week -- date )
6  7 \ DUP ( Y M D ) \R
7  GREEDAY \ WEKDAY \ SWAP -- ;
8
9 DAYS/MONTH ( S year month -- days )
10  2DUP 1+ 1 GREEDAY \ SWAP 1 GREEDAY \ D- DROP ;
11
12 WEKDATES ( S year month day -- )
13  \R \ DAYS/MONTH \R \ ( lastday day )
14  DUP 7 \ SWAP DO \ DUP 1 \ ROT BETWEEN IF
15  I 4 \R ELSE 4 SPACES THEN LOOP \ DROP ;
16
3 0 \ perpetual calendar
1  .DAYS \ " Sun Mon Tue Wed Thu Fri Sat" ;
2
3 .MONTH ( S year month -- )
4  CR 2DUP \ 1/ \ CR \ DAYS CR 6 0 DO
5  2DUP 2DUP 1 NTH-SUNDAY \ WEKDATES CR \ LOOP 2DROP ;
6 .YEAR ( S year -- )
7  DUP \ CR 13 \ DO \ I 16 \R I 1+32 \R I 2+32 \R CR
8  \DAYS 4 SPACES \DAYS 4 SPACES \DAYS CR 6 0 DO \ ( for each week)
9  3 \ J 1+ \ DO \ ( each month across page )
10  \DUP 1 2DUP J NTH-SUNDAY \ WEKDATES \ LOOP CR \ ( end of line )
11  \ LOOP \ ( end of weeks )
12  3 \ LOOP \ ( end of three month set )
13 DROP ;
```
**FUTURE86™**

The Language

**Finally, A Rommable, High Level Language That Puts You In Complete Control!**

A World-Class Language with all these Outstanding and Advanced Features:

- Produces extremely compact code with minimal overhead (less than 7kbyte min. COM file). Much smaller applications can easily be generated by customizing the kernel source code. (optional).
- Very fast execution speed — Assembler speed if desired.
- Best object readability of any high level language — Very desirable for real-time applications.
- Complete Assembly language support including 80186/8 instructions.
- Instantly mix Assembly language and high level statements in the same procedure without complex mode changes or awkward hooks.
- Inherently recursive and reentrant.
- DOS ERRORLEVEL support.
- Unique and powerful string handling support.
- 32-Bit addressing — Applications up to 64K code and data — an enormous FUTURE86 application. ROMMED applications can be much larger!
- Run other programs from within your program.
- More than 500 integrated, useful library procedures for easy application generation.
- Environment contains complete interactive symbolic debugging facilities.

**FUTURE86 is a mature language — already hard at work in thousands of diverse applications — worldwide! ... and it is still being extended.**

FUTURE86 is a compact, extensible language that lets you easily define your own high performance super language that is oriented to your application needs.

**FUTURE86 also includes these features:**

- Compiler compiles to machine code — No slow pseudo code interpreter.
- Conditional compilation resources and “INCLUDE” statement for maximum versatility.
- XENIX-like file access support.
- Fast two-pass compiler allows forward symbolic referencing.
- English-like commands without complex and cryptic syntax requirements typical of "C" or "Pascal".
- Inherently structured for easy readability — self documenting code!
- Extensive compiler and debugger error messages.
- Complete with efficient and versatile linker.
- Complete with comprehensive tutorial style manual containing complete procedure definitions and demo files that you can modify/compile/debug and run!
- No royalty for turnkey applications.
- Full technical support available.
- Full featured editor/word processor included.

**Unique in all the world!**

*For MS/DOS Machines*

Successful application examples:
- Automatic Postal Scale.
- Natural language compiler.
- Food processing & handling equipment.
- Family of multiuser terminal emulators.
- High speed Kanji character printer.
- Bar code and label software.

Soon available ... libraries for graphics and Hayes compatible protocol.

FUTURE86 contains everything you need to create your applications in a satisfying, creative, time-efficient manner. FUTURE86 is a fully-supported professional tool for serious applications.

**FUTURE86 — The best software investment you will ever make — Invest in your FUTURE today!**

Base System price (includes FUTURE86 Compiler, Kernel, Debugger, Editor, Demo files and manual) .......... $349.00
Source code for Kernel ........................................... $ 99.00
Source code for Debugger .......................................... $ 99.00
8087 Floating Point Support (includes source) ............ $ 99.00
Software Floating Point Support (includes source) ........ $ 99.00
RS232 Library (includes source) ............................... $ 99.00
Modem Library (includes source)* ........................... $ 99.00
GENHEX .COM file to Intel Hex file converter 
(includes source) .................................................. $ 99.00
One year update support ......................................... $ 99.00

*RS232 Library is a prerequisite

(All items include documentation)
(All prices are F.O.B. our factory)

**Special Bundled System Price — Introductory Offer:**
Includes all modules listed above — an $1,141 value for only $375.
Including one year update and technical support. Please note that this is a very Limited offer subject to withdrawal at any time.

To order or for more information call or write today:

**DEVELOPMENT ASSOCIATES**

1520 S. Lyon ..........................
Santa Ana, CA 92705 ..........................
(714) 835-9512 Community: 71460,1146

Future86 is a trademark of RIGY Corporation
© 1987 RIGY Corporation © 1987 DEVELOPMENT ASSOCIATES
© MS/DOS is a trademark of Microsoft Corp.
The ease with which the base for number conversion is changed in Forth can be used to advantage in many applications. In the code listed below, this feature of Forth is at the heart of a drill-and-practice system for binary number conversion. It was used by students in an introductory digital design course. In addition to giving the students practice in number conversion, it gave me a chance to illustrate Forth to them. The application was originally written in fig-FORTH on an AIM 65 microcomputer, but has been rewritten in F83 for presentation in this article.

Lines 13 and 15 of screen 3 illustrate the basic idea used to implement the system. Variables OUT-BASE and IN-BASE hold the bases to be used for interpreting output and input data, respectively. QUIZ administers the practice sessions by presenting numbers using OUT-BASE data as base, and interpreting answers using IN-BASE data as base, until five correct responses have been given.

The numbers needed for the practice sessions are derived by means of a random number generator defined in screen 1, lines 10 and 11 (see Burton, "The Game of Reverse," FD III/5). SET-UP presents a number to the student, GET-ANSWER receives the student's response, and TEST-IT evaluates the response. Because the original version was administered using the AIM 65 (with its single-line display), the quiz prompts were kept simple. One change that might make the drill system more user friendly would be to display the two reference bases as reminders for the student, and a count of correct answers.

Of course, the really neat thing about this code, typical of many Forth applications, is the ease with which variations can be generated. Screen 4 gives several more quizzes, including two for conversion between hexadecimal and binary, and one from base three to base five (I dare you to try that one!). Some examples of the program dialog are included, too. In the examples, underlined information is the program output, the rest is the student's response.

Richard H. Turpin, Ph.D., is professor of Electrical and Computer Engineering at the University of the Pacific.
Drill and Practice in Number Conversion

by Richard H. Turpin
Prof. of Elec. Engr.
Univ. of the Pacific
Stockton, CA 95211

VARIABLE IN-BASE
VARIABLE OUT-BASE
VARIABLE SEED
CR . ( Type any key, please.) CR KEY SEED !

RND . (n -- n)
SEED @ 259 # 3 + 32767 AND DUP SEED ! 32767 #/;
GET-A-NUMBER ( -- n ) 256 RND ;
INPUT ( -- d,a ) PAD 20 0 FILL
PAD 10 EXPECT 0 0 PAD 1- CONVERT ;

WHAT 2DROP CR . WHAT? ;
BEGIN INPUT CR WHILE "WHAT REPEAT DROP ;
GET-ANSWER (-- n ) IN-BASE @ BASE ! RESPONSE ;
SHOW-IT ( n -- ) CR OUT-BASE @ BASE ! 8 . R 3 SPACES ;
CORRECT ( c,n -- c+1 ) CR DROP 1+ ." CORRECT " ;
INCORRECT ( n -- ) CR ." ANSWER IS " .;

GET-ANSWER (-- n ) IN-BASE @ BASE ! RESPONSE ;
OVER = IF CORRECT ELSE INCORRECT THEN ;
CONGRATS CR ." 5 CORRECT . CR ." GOOD WORK! " ;
SET-UP ( -- n ) GET-A-NUMBER DUP SHOW-IT ;
QUESTION ( c -- c' ) SET-UP GET-ANSWER TEST-IT ;
QUIZ 0 BEGIN QUESTION DUP S = UNTIL DROP
DECIMAL CONGRATS ;
DECIMAL.TO.BINARY 10 OUT-BASE ! 2 IN-BASE ! QUIZ ;
BINARY.TO.DECIMAL 2 OUT-BASE ! 10 IN-BASE ! QUIZ ; -->

(Continued on page 36.)
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. 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.

RESERVATIONS FOR NINTH FORML CONFERENCE

Registration fees for conference attendees includes conference registration, coffee breaks, and notebook of papers submitted, and for everyone rooms Friday and Saturday, all meals from lunch Friday through lunch Saturday, wine and cheese parties Friday and Saturday nights, and use of Asilomar facilities.

Conference attendee in double room - $275 • Non-conference guest in same room - $150 • Children under 17 in same room - $100 • Infants under 2 years old in same room - free • Conference attendee in single room - $325

Send reservation requests to:

FORML Conference, Forth Interest Group, P. O. Box 8231, San Jose, CA 95155
A major advantage of computer-assisted instruction (CAI) is that lessons may be repeated as often as the learner wishes, or until some prescribed “mastery level” has been attained. One of the methods used in CAI to improve retention of material is to quiz the learner on what has been presented. Since the lesson can be repeated, to be effective a quiz should not ask the same questions in the same order every time. This paper describes a simple method for thoroughly “scrambling” a multiple-choice, matching CAI quiz.

Currently, I am developing a computer-assisted instruction tutorial on highway superelevation for my class in landscape architectural construction. In the quiz module, I included questions to identify forces acting on a vehicle in a curve by name and by formula. One question tags elements of a force vector diagram by letter, and lists element names by number; another uses the same diagram with a numbered list of formulae. Each question consists of eleven matching identifications: “What is letter a?” etc. The student enters the corresponding number of the name or formula; the program checks the response, gives feedback, shows the number of the correct response, and waits for a keypress to continue. Figure One shows a typical display.

Obviously, though, if students repeat the identical quiz several times, some will remember that “the answer for a is 7,” rather than learn the correct name and formula for each element. Therefore, I wanted to randomly arrange the items presented so that on the first try, a might be “centrifugal force,” and the next time a might be “weight force normal to the slope,” and so forth. The problem was how to ask the questions in alphabetical order, yet randomly assign letters to the diagram components while keeping track of the correct answers.

First, I sketched the diagram on paper and listed the names of the elements. Next, I numbered the list of names and somewhat arbitrarily assigned a letter to each element on the diagram. a was “7.θ (theta),” b was “3. Force parallel to the slope,” etc. The relationships originally established were as shown in Figure Two.

Suppose we shuffle the letters of the elements. If we tag the diagram in the same order, the position of each letter both on the display and in the list of elements will still directly correspond to its answer in the list of names. We can tag the three 8’s (sevens) with three different letters, but they are still 8’s. (θ by any other letter....) Matching the character’s position in the element list will point to the answer in the Name# list. Matchpoint!

Remember, the numbers used to identify the names and formulae are not shuffled, and the diagram does not change. Only the identifying letter tagging each element of the diagram changes. Changing only the order of letters in the element list changes both the letter used to tag each element on the displayed diagram and, because the questions are always asked in alphabetical order, the order in which the student is asked to identify any given element.

About the Code

Screens 0 through 9 contain definitions to run a facsimile of the matching questions used in the quiz. The tutorial is written in LMI’s PC/Forth 3.1 and requires both EGA graphics and floating-point extensions. The definitions listed here do require LMI’s EGA graphics but not the floating-point extensions; the method itself requires neither.

Screen 1 contains several words to simplify formatting, and screen 2 builds the arrays. LMI’s word “,” (comma-quote) creates an uncounted ASCII string, A$. One could also C, (C-comma) the numerical ASCII values of the first eleven lower-case letters after CREATING A$. ANSWERS holds the “correct answer” numbers corresponding to the letters in A$. CSWAP and SHUFFLE, from Thinking Forth, Appendix D, “shuffle” A$’s alphabetical order using the random number routines found in Starting Forth (pg. 265).

Once A$ has been shuffled, the order of the characters is random, but each character’s position in the shuffled string is still directly associated with the numeric answer stored at the corresponding position in ANSWERS. In other words, we don’t care what letter is at position 3 in A$. The letter at position 3 in A$ will be displayed on the diagram to tag the same element every time. The correct answer for any character tagging that element will always be 2 because 2 is stored in ANSWERS at position 3. Figure Three illustrates how this works.

In the shuffled A$ in Figure Three, a has replaced j in the original A$. To answer the first question, “What is letter a?” we locate the position of a in A$ (in bytes). The corresponding answer for that position is stored at 2* that number of bytes from
ANSWERS, so fetching the correct number from ANSWERS is all that’s required. We match (find) the character which points to the answer. a is located nine bytes “downstring” from A$L, so the corresponding answer is located nine cells “downnumber” from ANSWERS, in this case the number 9. b is located where a was located in the original A$L, so next time through the loop, the correct answer for “What is the letter b?” will be 7 (position 0 in A$L, 2*, ANSWERS, +).

Skipping to screen 9, FINDCHAR takes the index of the loop in GIVEQUIZ and adds 97 (ASCII “a”) to put the ASCII value of the ith letter of the alphabet on the stack. It then puts A$L’s address and 11 (A$L’s length) on the stack, ROTs the character to the top, and SCANs A$L. SCAN (an LMI word) returns the address of the byte which contains the character and the number of bytes remaining in the string. We don’t care about the remaining bytes, so we DROP them, leaving the character’s address. Subtracting A$L’s address leaves the offset in bytes, the position of the letter in A$L. (If you don’t have SCAN, see Michael Ham’s “Wordwrapping Tool” (Forth Dimensions VII/4).

MATCHANSWER then takes the position (bytes) that FINDCHAR leaves on the stack, multiplies by two (bytes to cells, remember), adds ANSWER’s address and fetches the correct numeric answer. It is that simple!

Screen 8 contains simple definitions for answer processing to help explain GIVEQUIZ in screen 9. GIVEQUIZ loops through ASCII a–j, incrementing the character, finding its position in A$L, finding the corresponding position in ANSWERS, and evaluating the student’s response. Because the loop’s i is used to calculate the ASCII value of the letter of the element being asked, the questions are asked in alphabetical order.

Screens 3 through 7 contain words required to display the problem. In screen 3, the 2VARIABLE CP holds the current cursor position. DRAW draws from CP to X,Y on the stack, and updates CP to the endpoint; AT resets CP to a new location.

SMALLFONT and NORMFONT use LMI’s ROM BIOS call video-io to select one of the two resident EGA fonts. The font routine is not “proper,” in that IBM says the call should be used only immediately following a “mode set” which clears the screen. Using the routines shown allows the use of both fonts on the screen at the same time. It may not be “proper,” but in graphics mode, it works!

Screen 4 definitions draw the diagram, and TAG’EM in screen 5 tags the elements.
THE IPC-SBC38 DEVELPOMENT AND CONTROL SYSTEM

WRITE IT — RUN IT — ROM IT

A single board computer development and control system that is simple to use, you will be developing applications programs the first day:
- Choice of Basic or FORTH
- In ROM
- 8 channel 8 bit analog to digital converter
- Two 8 bit input ports
- Two 8 bit output ports
- Time of day
- ROM 16 bit addressable
- Low Cost: $99 in 1988

MasterCard and Visa accepted

Vesta Technology, Inc. 7100 W. 44th Ave. Suite 101 Wheat Ridge, CO 80033 (303) 422-8088

OEM188 SBC DEVELOPMENT SYSTEM FOR PRODUCT APPLICATIONS

The OEM188 - designed to bring your product to market in the fastest possible time - through the most productive software development environment available & cost effective hardware.
- The OEM188 boots MS-DOS or CP/M-86. Write your program in Assembler, Fort, Basic, C, Fortran or Pascal.
- ROM your code. The EPROM programmer is onboard and fully integrated into the hardware and software.
- Develop your code quickly with Vesta's ROMmed languages designed for control tasks.
- Size 8" x 8" FDC for 4 drives, Dual UART with RS-232, TTL and RS-422 I/O, Bus - IBM, Printer port. Watchdog, Battery backed real time clock and up to 256 K static RAM/ROM. Programmer interface - terminal. Various I/O boards available.
- Prices starting as low as $329 each.

V ESTA TECHNOLOGY, INC.*7100 W. 44th Ave.*Suite 101
Wheatridge, CO 80033 * (303)422-8088 * VISA & MC

CUSTOM DESIGN FOR CONTROL COMPUTERS

Why put this expensive specialist on your payroll when we already have the support you need? Vesta specializes in the design and production of control computers for OEM's.
- CREDIBILITY We have thousands of installed systems
- SUPPORT Your application engineer is as close as your phone
- ECONOMICAL Your break even quantity may be as low as 50 units
- FAST Typically 3 to 4 months from ideas to fully functional product
- PROGRAMMING From BIOS to application level
- PRODUCTION If you don't want to, we do

Let us help you get your product into the market, ahead of your competition and at a reduced cost. Call us to discuss your requirements. Our prompt quote will make you happy you did.

VESTA TECHNOLOGY, INC.
7100 W. 44th Ave.*Suite 101*Wheatridge, CO 80033
(303)422-8088

Diagram Formulae:
1. \( \frac{Wv^2}{gR} \)
2. \( Wv^2 \sin \theta \div gR \)
3. \( Wv^2 \cos \theta \div gR \)
4. \( W \)
5. \( W \cos \theta \)
6. \( W \sin \theta \)
7. \( \text{arctan} \ e \)
8. Center of Gravity
9. rise per foot

What is letter b?
3
WRONG
The correct answer is 2

Press a key to continue.

Figure One. Typical CAI display.
The definitions in screen 8 are simplified "cousins" of the answer-processing routines actually used. Because the rest of the quiz is mathematical problem-solving, the "real" GET-ANS uses a modified, floating-point variation of Michael Ham's "Write Like a Fox" numeric-entry routine (FD VI/3). CHECK-ANS chooses a random feedback string instead of just displaying "right" or "wrong," etc., but the tutorial isn't the point here. The incredibly simple technique in the example screens produces two matching problems consisting of twenty-two thoroughly scrambled questions with evaluated answers. That's Forth!

(You may add your own scoring routines...)

A Sidebar for the Interested

The superelevation tutorial which spawned this article is currently eighty-some screens of definitions which compile to a 56K LMI turnkey. This autumn, the tutorial is being used in the data-gathering phase of a doctoral dissertation, Examination of the Cognitive Style Construct Field-Dependent/Independent as a Student Selection Criterion in Formative Evaluation. In developing CAI, as in developing any program, user feedback is useful. The "formative evaluation" will result in two modified tutorials: one based on feedback from field-dependent students, and one based on feedback from field-independent students. All three forms of the tutorial will be used to analyze the effectiveness of feedback from students with different learning styles on the development of learning materials.

Figure Two. Original order of elements and names.

Element: a b c d e f g h i j k
Name#: 7 3 1 2 8 7 6 4 5 9 7
(Note that 7 is the correct answer for three items because the angle, \( \theta \), occurs three times.)

Figure Three. Shuffled A$ contents correlate to correct ANSWERS via their position.
This program as written, requires LMI's EGAGRAPH.COM graphics driver be loaded first, or you will crash.

Written in Laboratory Microsystems, Inc.'s PC/FORTH 3.1

(Continued on page 36.)
Dateline San Jose

FIG Now Accessible Via Modem

Dateline San Jose

The Forth Interest Group (FIG) is up and running on GEnie, the growing modem access communication network that is part of GE Information Services. GEnie is a nationwide service that is also available in Canada and Japan, with plans for expansion into Western Europe. You can now communicate with other Forth users, and download software and commentary from the FIG sponsored Forth RoundTable on GEnie. Type "FORTH" at any main menu prompt to get to the FIG area.

The FIG area on GEnie has three main areas that are, in turn, divided into several smaller categories. The three main areas are:

1. Bulletin Board - A message oriented information exchange with the following categories:
   - Introduction to FIG Online
   - Introduction to Forth
   - Techniques and Tutorials
   - Product Announcements
   - Calendar of Events
   - Applications in Forth
   - Forth Dimensions Feedback
   - FIG Chapter Topics
   - FIG Order Line
   - Forth Standards
   - The Job Market
   - Questions and Answers

2. Live RoundTable - Discussion area where pre-announced, interactive discussions may take place. There are three separate areas for people to gather and exchange information or chitchat.

3. File Area - A large selection of software and commentary files are available for downloading.

FIG members who sign up using the special FIG account number at the regular $18 one-time enrollment fee will get three free hours of time on GEnie (6 P.M. to 8 A.M.). After the initial three hours, all subsequent time is billed at the usual GEnie rate of $5 per hour during non-prime-time hours. This is one of the lowest rates around for an information service. Be warned, though, that the 8 A.M. to 6 P.M. rate is $35 per hour! So watch your online times to avoid unwanted prime time charges.

To join GEnie, use your modem to call 1-800-638-8369 with your communication parameters set at no parity, 8 data bits and 1 stop bit. Use 300 or 1200 baud setting and half duplex (echo on). Following connection, type HHH (without CR). Then, at the U# prompt, enter XJM11849,GENIE (CR) to sign on. This will put you at a menu where you may have a demonstration of GEnie, get your local telephone access number (node), and billing information. You will have the option to sign up for GEnie. Have your VISA, Mastercard, Discover, or American Express card handy or have the initial subscription fee electronically transferred from your checking account by using the number on your checks.

Next you will have to answer a few questions (name, address, etc.), and electronically agree to the terms of the service agreement. For immediate access, use a credit card. GEnie will issue a password and provide the phone number for your local GEnie access node which will allow you to use the GEnie system. You will receive your user's manual and other information shortly.

The GEnie customer service (voice) number is 1-800-638-9636 if you have any problems. Service is available from 9 A.M. to 1 A.M. weekdays and noon to 8 P.M. weekends and holidays. (Eastern Time).
We all need tools, sometimes to take things apart, sometimes to put things together. Of course, this is true of programmers just as much as it is of mechanics, cooks, and burglars. Forth makes both the design and utilization of programming tools easy, letting each one of us establish a programming environment that suits our needs and personalities.

This article will give you some tools to help you explore your Forth environment. The main word (tool) I present is called SEE; it is a high-level Forth decompiler. Many versions of Forth already contain a decompiler, but mine, in addition to the source code, displays the addresses for each element. This enables detailed exploration of the dictionary structure, and it gives all the information necessary to perform seat-of-the-pants patches. It can be compared in power with either a sledgehammer or a jeweler's screwdriver, depending on how you use it. Sometimes both are necessary.

You probably realize that one of the main differences between Forth and traditional languages is the number of intelligent words each language contains. Simply, every word in Forth is intelligent. It contains within itself a complete description of what it does and who it is. Compare this to language X, wherein every word is void, a mere character string, with the exception of the compiler or interpreter itself. Language X is this compiler or interpreter, unchanging, everlasting, dull. Since a Forth word knows itself, why not just ask it to share this information? This is what we are going to do.

My Forth decompiler, SEE, takes a word and divides it into all of its components. It then displays this information in an organized manner which is ideal for finding patch locations, investigating how a word works, and acquiring considerable knowledge on how Forth is put together. If you have already completed Forth 101, skip ahead to the section entitled, "The Visible and Invisible Tools" — the rest of us are going to investigate what these component parts are. Our initial breakdown of a Forth word will be into two parts, a header and a body.

The **Id** of Forth

The header is what gives Forth its interactive power. In a turnkey Forth application which has been scrutinized down to the bare minimum necessary to accomplish the application, the header is not even necessary. What you have left, however, will not "feel" like Forth; it will be a word processor (only), or a video game (only), or a spreadsheet (only). They may, indeed, be very good word processors or video games, but one advantage of Forth is that it bypasses the "only's" given by other languages. A header is used by the outer interpreter when compiling new words into the dictionary or interpreting commands from the keyboard. It is not used during the actual execution of a word and, hence, has nothing to do with the speed of execution, no matter how big the header is. The Forth header is subdivided into two parts, or "fields."

**Where Am I?** The second part of the header is the link field. This is just one cell (16 bits) wide. When the outer interpreter is hunting for a given word in the dictionary, it starts at the top (LATEST word) of the dictionary and checks if it is the word it is looking for. If not, it goes to the link field to discover where the name field of the previous word in the dictionary begins. SMUDGED words are ignored. This continues until either the desired word is found or the link field contains zero, which would announce the bottom of the dictionary and an unsuccessful search. Note that the trend in newer versions of Forth is to place the link field immediately prior to the name field, a format which allows quicker dic-
tionary searches. If your version does this, a simple modification to my code can be made.

So there is the header, both name field and link field. The body is also divided into two fields, the code field and the parameter field. The body is required, even in the minimal turnkey system mentioned above. It is what makes the Forth word intelligent.

What Do I Do? The code field is one cell wide, commencing immediately after the link field. It is a pointer, and it always points to machine language code (it took this a while to get through my thick head). This code may be for colon definitions, variables, constants, CODE definitions, or whatever other animals may be in your system, but it is always low-level, machine-executable code. Sometimes, as in the case of CODE definitions, this code immediately follows the code field. Usually, however, it is a very short routine explaining how to handle the parameter(s) of the parameter field. For example, if I am a CONSTANT, I take the next cell after the code field and push its contents onto the stack. I am then through, my mission completed. NEXT.

What (or Who) Do I Do It With? The parameter field is made up of as many cells as are necessary, from zero up. In some cases, such as for constants and variables, these are parameters in the sense that other languages use the term. For the common case of colon definitions, however, we have very special parameters, a list of cells containing the code field addresses of the words used to define this word.

Hold on tight, I am going to try to confuse you. Above I said that the code field always points to machine code. However, where the code field of a given word is located, its "code field address," is not machine code. Thus, in this list created by a colon definition, none of the code field addresses point directly to low-level code. With the decompiler, you are going to see things that seem very strange until you understand this concept.

Additionally, sometimes the decompiler will display what I call "invisible words." Not many of us have actually used `OBRANCH or `LET in our definitions, but when we use IF or a literal, they magically appear. As we think about what has to happen, the reason for these invisible words

```
scr # 2
0 ( Miscellaneous core words and modifications)  DECIMAL
1
2 243 VARIABLE MAXSCR ( will be 387 on non-system disk )
3 : LOAD ( scr#) DUP MAXSCR @ > b= 4 IF LOAD DECIMAL ELSE 6 ERROR ENIF ;
5 : \ ( skip rest of line)
6 : IN @ 64 / 1+ 64 * IN @ ; IMMEDIATE
7 : THRU ( n1 n2) \ load screens n1 thru n2 inclusive
8 1+ SWAP DO I LOAD LOOP ;
9 CREATE BYE HEX OC3 C , , SMUDGE DECIMAL
10
11 : CCONSTANT <BUILDS C, DOES> C0 ;
12 16 CCONSTANT 16 4 CCONSTANT 4
13 14
15 ;S

scr # 3
0 \ .words to aid programmer
1
2 : .D ( n) BASE @ SWAP DECIMAL . BASE @ ;
3 : .VOC CONTEXT @ @ . NFA ID . ;
4 : .DEF CURRENT @ @ . NFA ID . ;
5 : @ R . DO @ LOOP @ > TYPE ; \ .R with @ fill
6 : .BASE .BASE @ .D .;
7 : .H ( n) BASE @ SWAP HEX U. BASE @ ;
8 : .B ( n) BASE @ SWAP @ 16 2 BASE @ .R SPACE BASE @ ;
9 : S \ nondestructively print data stack
10 SP@ 50 @ SWAP 2D UP = IF 1 MESSAGE DROP DROP
11 ELSE DO I @ . 2 +LOOP ENIF ;
12 : .FRE SP@ HERE - .H .;
13 : U? @ U. ; \ "U-query"
14
15 ;S

scr # 7
0 \ Non-VOCABULARY extensions
1
2 : -QUIT ?TERMINAL IF KEY 3 = IF QUIT THEN THEN ;
3 : BETWEEN ( n1 n2 n3 -- f) \ check n1 between (n2,n3)
4 : ROT DUP ROT @ >R < R> AND .
5 : "EMIT \ emit only ascii between 32 and 127 else emit .
6 : DUP 31 128 BETWEEN IF EMIT ELSE DROP ." . ENDIF ;
7 : PAD2 ( --a) PAD C/L + ;
8
9 ;S
10
11
12
13
14
15

scr # 8
0 \ DUMP, hex and ascii
1
2 : BUNCH ( n1 n2) MOD @ = IF SPACE ENDIF ;
3 : (DUMP) ( 1o hi) CR
4 1+ SWAP DO 1 DUP @ 4 .R 5B EMIT SPACE
5 16 .R 0 4B 14 BUNCH 1 + C@
6 1 2 .R 0 4B 14 BUNCH C+ C! LOOP
7 2 .R 0 4B 14 BUNCH C+ C! LOOP
8 CR -QUIT DROP 16 +LOOP ;
9 : HEADDUMP CR " addr " OVER 17 1 DO
10 .R DUP @ 16/ DROP . I 4 BUNCH 1 + LOOP DROP CR ;
11 : DUMP BASE @ >R HEX HEADDUMP (DUMP) R> BASE @ ;
12
13 ;S
14
15
```
becomes clear. For example, if our colon definition contains the literal 5783, we need a filter to say, “Hey, Inner Interpreter, the next cell is not an address of a code field; it is a number. Just let it be pushed onto the stack when we are executing.” This filter is the word LIT. Many immediate words, such as IF, in addition to doing something during compilation, compile invisible words such as 0BRANCH followed, in this case, by how many cells to skip when the value on top of the stack (during execution) is zero.

This concludes my Forth 101 lesson. Note that other people divide Forth words in different ways, sometimes using different terms. Understanding the above will help you to comprehend someone else’s explanation. I hope this background information will help you to appreciate the beauty and simplicity of Forth, which you will find with SEE.

The Visible and Invisible Tools

Now we can start talking about the tools to make your version of Forth visible. Like almost any tool, SEE requires other tools to build it. A useful one, even apart from SEE, is a DUMP word that displays both hex and ASCII. Every version of Forth I have seen contains some sort of memory DUMP word, but they do not always show both hex and ASCII. Sometimes (shiver), they are merely decimal dumps. This is a “personality” word; when you call it, do you give it the starting address and a count to dump, or do you wish to give it the starting and ending addresses? The choice is entirely yours. The version I wrote uses the latter approach (I was used to it, since my ROM monitor uses this method). If you prefer the former method, merely add this redefinition on line 12 of screen 8:

```
: DUMP ( addr count — )
  OVER + DUMP ;
```

This version is formatted for a 64-column screen and was modelled after the D command of the excellent CP/M public-domain disk utility DUPM.

Note that some of the words used here are from screen 7. The nicest of these is -QUIT (“don’t quit”), which normally does nothing (it doesn’t quit). If it senses a
keypress, though, it pauses the listing until another key is pressed. If either the initial or the second keypress is ^C, it performs a QUIT; otherwise, the second keypress continues the display. Other keyboard monitors (than that of an Exidy Sorcerer) may react slightly differently, since my ^TERMINAL word goes directly to a special ROM routine rather than to the normal keyboard scan. If the definition does not work on your computer to your satisfaction, play with it until it does. It is a very useful word for which you will find many applications. A substitute, which will merely cause a QUIT if any key is pressed, may be made. It is not as nice, but it is still useful, and it should work with any kind of keyboard scan:

```
: -QUIT ?TERMINAL IF QUIT THEN ;
```

Since I use many tool-building tools, I have included screens 2, 3, and 9. Screen 2 contains some suggestions found in Leo Brodie’s Thinking Forth. Particularly useful is the word \, which allows a line comment (similar to ; in many assemblers). Note that the definition for LOAD automatically resets BASE to DECIMAL after the LOAD (the other stuff surrounding LOAD is peculiar to my disk configuration). The only words here necessary for SEE are 0 .R (i.e., .R with zero fill) and .H (often called H., which would be inconsistent, since .D. is already taken, and .D. does not force a decimal . of the stack). This is mainly a group of words to let me know what my present system configuration is; I tend to change bases more often than a softball player. Use what you like; after all, it is your Forth. Screen 9 is an improvement to Dr. Charles Eaker’s CASE statement. This is used for the different CASES of exceptions.

Finally, I am ready to discuss the tool for which you are reading this article. The SEE tool presented here occupies three screens (16 - 18). Screen 16 should work without significant modification on any version of the FIG Model (however, on cell address machines, change the 2+ on line 2 to a 1+, and change the 2 on line 13 to a 1). There are two if’s: If you choose a DUMP definition requiring address/count, on line 3 delete the words DUP and +. If your LOAD word does not automatically do so, on line 6 reset BASE to DECIMAL after loading the exceptions screens (or redefine LOAD). Only minor modifications are necessary on other versions; just do the things you have to do in converting any program. That may not be much help, but all the logic is there. I have adapted it to three systems, one of them pre-fig-FORTH, using this logic. Since you are a Forth programmer, I have saved a little bit of work for you. I promise the work will be worthwhile; merely adapting the program to your system will, in itself, teach you much about Forth. Screens 17 and 18 will require some interactive play on your part. The addresses contained therein will be correct for an unmodified 8080 fig-FORTH (mine is modified, but I kept the starting address of everything the same), but for most of you, some address changes will need to be made.

The two initial words on screen 17 make the handling of invisible words simpler and improve the display. Then comes EXCEPTIONS itself, which is merely a series of cases. If the address contained on top of the stack (when EXCEPTIONS is called) matches one of the exceptions, that code is executed and at least one zero flag is returned to see (two zero flags lets see know that it is through). If there is no match, the non-zero address (interpreted as TRUE) is returned, which causes see to treat it as if it were a normal code field address, which it will be 90% of the time. The word SNAME on screen 16 is the heart of see, assuming there are no exceptions. It merely takes a code field address from the top of the stack and prints the name of the word which has that CFA.

Still, 10% of the cases must be handled separately, so we have to adapt EXCEPTIONS for those cases, and with your addresses. The most straightforward method here is brute force. Start out with a null version of EXCEPTIONS(: EXCEPTIONS DROP ;). Then see words whose definitions you already understand pretty well; like see HERE (the fig-FORTH definition is : HERE DP @ ;). Immediately before it starts printing garbage, an address will be displayed which will be the exception address needed. (Remember that -QUIT will allow you to break at any point

---

**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
- DECOMPILERS 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
by typing ^C.) On the first run of \texttt{SEE} here, this will be the address for: (which is not an ordinary definition in fig-FORTH, so in my exception I print the : directly, \texttt{DROP} the CFA, and return zero).

After you have made this modification to \texttt{EXCEPTIONS} (get rid of the \texttt{DROP} that was the initial definition!), run \texttt{SEE} here again. Now it will perform beautifully all the way to the ;. When it prints the subsequent garbage, it will already have printed both the address of ; and the one-character string ; (this is an ordinary fig-FORTH definition, but it needs to be an exception so see can know when to stop). Now add the ; exception per the model on line 5 of screen 17. Non-fig-FORTH models may not act normally with the ; — the way to tell is to note if a ; and \texttt{CR} were printed out. If they were, the ; was treated normally. If not, treat the exception in much the same way: was treated on line 4, only follow the \texttt{DROP} by two zeroes.

Instead, remember that you are making your version of Forth visible to you; the more you learn about it, the greater its visibility will be.

There is a more orderly approach you may take in this adaptation; it will not save any time, but it will prevent gross hacking. You have a tool I did not have, the word \texttt{ISEE} ("Initializing SEE"). For the right-most words on screens 17 and 18, merely type \texttt{ISEE} \texttt{;} or \texttt{ISEE} \texttt{0BRANCH}. Three lines will be typed. The bottom line will be the link field, which is contained in the address to its left. The following cell after this address (two bytes, or one cell, depending on your system) will be the code field, which will usually be the exception address you were looking for. In the cases where it is not, such as \texttt{;}, do an \texttt{ISEE} of a colon definition (in this case), as in \texttt{ISEE HERE}. Then do a \texttt{DUMP} using the left-most address on the top line as your starting address, and about 64 higher than that for your ending address.

You will discover two exceptions upon examination of this dump, for \texttt{;} and for \texttt{;;}. The way to tell which method you will need (at least in a fig-FORTH model) is to see if my exception pattern contains the word \texttt{SNAME}. If it does, the first method may be used; if not, use the latter. Note that the latter method will always work, but may give you more information that you will immediately understand.

Finally, let us look at some actual \texttt{SEE} decompositions from my system. I have them fully commented in Figures One and Two. I tried to choose samples of many of the things you will run into. The comments are in a different typeface than the actual decomposition, and this method may be used to get a two-dimensional understanding of the way a given word works. In each sample, three fields per line result from the decomposition. The first field is the address of what follows on that line. In some cases, such as compiled strings or many of the invisible words, this is just the first address; listing all the addresses would be useless, since they follow linearly. The second field is the contents of this address, usually the CFA of another word. The third field is just the name of this word pointed to, but may be a compiled string or some other excepted animal.

Note that this does not look like a source definition is expected to look. It is not supposed to. I organized it to contain all information the word knows about itself, more than was included in the source definition. \texttt{IF} has become \texttt{0BRANCH}, \texttt{ELSE} has become \texttt{BRANCH}, and \texttt{THEN} and \texttt{BEGIN} have disappeared entirely. There will be other surprises as well, but I promised you a visible Forth, one showing what your Forth is, not what you imagined it to be. You will discover it is better than you imagined.

I will comment briefly on \texttt{CODE} definitions. My listings merely print a \texttt{DUMP} when machine code is sensed. While a \texttt{Z80} (or 6502 or 8086 or PDP-11) disassembler is quite practical, it is not a tool I have found necessary to build. Most machine code used in Forth is quite short and easy to disassemble by hand, when desired. It is your Forth; if you want this tool, build it. It can be patched into \texttt{SEE} by replacing your disassembler word with my word \texttt{CDMP} (Code Dump). Then share it with the rest of us — it is our Forth, too.

\textbf{Endnotes}

1. There are rare exceptions. I have seen a Forth calendar containing words for the twelve months, that uses the text within the headers to print out the name of the desired month. A Forth programmer can use any part of Forth in any manner desired.

2. Consider that the vast majority of string comparisons during a search return \texttt{FALSE}.

(Text continued on page 37.)
Length byte (true length found by ANDing with 1F)
ASCII name (the max. screen number on my drive)
Link field
Parameter field (for formatting purposes, the value is included on the same line)

Characters per Line
Note that all numbers are shown as positive, 16-bit, hexadecimal characters

the EXCEPTION for USER automatically looks up the actual value, which had an index of 26 hex

VOCABULARY definition, built with <BUILDS ... DOES>
Note what DOES does. It compiles addresses into the dictionary, the first of which is where execution will be transferred. It also pushes the location of the second address of the compiled addresses onto the stack for utilization by the executable code. A perfectly remarkable GOTO in disguise!

Note how a CODE definition may be sensed (the 2 difference between 07F4 and 07F6). The machine code ends at location 07FB. Try to decompile HERE by hand. Then get busy and type in "SEE"!

Figure One. Miscellaneous forms. Note the three columns per decompilation:
Location, Contents of Location, and Description.
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

TECHNICAL SESSIONS
Covering the latest in Forth technology

Conference & Exhibit Hours
Fri. Nov. 13 12 noon - 6 pm
Sat. Nov. 14 9 am - 5 pm

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

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

Preregistrations @ $20 ____ Banquet Tickets @ $35 ____ Total check to FIG (US funds) $________
Name ________________________________
Company ________________________________
Address ________________________________
City__________________ State ______ ZIP _______ Country ________________________________
Telephone (______) ________________________________

Return to: Forth Interest Group • P. O. Box 8629 • San Jose, CA 95155
...but what if I really, really want to begin with screen 16?

"jumping" EXCEPTIONS will automatically compute and show where they "jump" to

Ok, sometimes I'll do it the "standard" way.

Note that IBDS5 is the very first byte of 3LIST

Also note this link to 3LIST. The IBDD does not point to machine code, but the contents of IBDD do.

provided as a favor to those who tried to decompile HERE from the DUMP in FIGURE 1.
What's in the disk buffers?

# of disk BUFFers (24 in my system)

control loop --------------------------

Bytes per disk Buffer (256 in my system)

quicker than (LIT) 4 +

address of FIRST disk buffer

Figure Three. 2-Dimensional Word Analysis

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
These notes about the first meeting of the ANS Forth Technical Committee (August 3-4, 1987 at CBEMA Headquarters in Washington, DC) are not minutes of the meeting, nor are they part of the ANS Forth Technical Committee's official documentation. They represent my personal impressions only. The reason I mention this is that during the course of the meeting, it became clear there are serious liability considerations for anyone publishing anything which someone could interpret as anything approaching a standard. Therefore, I must caution everyone that this is NOT a standards document. With that out of the way, I'll try to give you a summary of the events.

The meeting began at 9:00 a.m. on Monday and was attended by many of the leaders of the Forth Community, including a number of folks from the Forth Standards Team (FST). Elizabeth Rather (as the "convener") was acting chairperson, and Ray Duncan volunteered to be acting secretary for the initial meeting. Elizabeth made some opening remarks on the maturation of Forth and the need for greater acceptance. She described the scope of this effort as mainly oriented towards describing common practice, neither attempting to fix Forth nor using the standard as an instrument for change.

The agenda was approved without dissent.

There followed a discussion on international involvement. The X3 representative suggested that early involvement would lengthen the process. It turns out that there is a requirement for an international liaison, but it wasn't clear at what point that would become important. Chuck Moore felt strongly that international involvement was important. I don't recall that any decision was reached on this.

Following this was a discussion on validation suites. The original scope proposal specifically excluded them from this group's activity. The membership voted not to exclude the possibility of such development.

Cathy Kachurik of CBEMA presented a tutorial on the X3 structure and process.

Charlie Keane proposed adopting the Forth-83 Standard as a "BASIS" document. As I understood it, the basic document becomes the working document for all activity of the technical committee (TC). That is, all changes, deletions, additions, etc. are proposed as updates to this document. There was a discussion about whether to restrict this document to Chapter 12 (the Required Word Set). This was defeated 15-1. A motion to adopt chapters 1-16 carried unanimously. This excluded only the appendices (uncontrolled reference words, experimental proposals, charter, membership, and proposal/comment forms).

I put up a motion to include floating point as part of the scope of work. This was defeated by a vote of 4-12. A motion was approved (14-3) to eliminate the time-frame constraints on standard language extensions. Previously, this had indicated that language extensions could not be considered until the approval of the ANS Forth standard.

Two additions to the Scope of Work document were approved: that the TC will review existing and proposed programming language standards; and to consider the impact of the standard on current and anticipated hardware technology. Another change to this document was to change the number of users required for a Forth system in order to be considered for non-compliance with the Forth-83 standard; this number was reduced from 1000 to 200 users of a particular Forth implementation. The Scope of Work document (X3J14/87-002) was then approved as amended.

The TC Subcommittees document (X3J14/87-004) was approved with minor changes. There are four subcommittees: Documentation, Research, Logistics, and Technical.

The Plan of Work document (X3J14/87-003) was then approved with a few changes. It was agreed to remove specific topic areas from the meeting plan in order to allow the work to proceed faster if possible. Chuck Moore got a motion passed to require that at least one meeting be held in San Francisco. The membership then agreed to hold the next meeting in San Francisco, but could not find anyone willing to host that meeting. Elizabeth Rather then volunteered to host the meeting in Southern California, and that was agreed on. I put up a motion to co-schedule and co-locate the ANS Forth meetings with the FORML and Rochester conferences. This was defeated 4-10.

Next was the Call for Officers. The X3 Secretariat appoints the officers from a list of volunteers. The following people volunteered at the meeting:

Chair: Charlie Keane, Bill Dress, Larry Forsley
Vice Chair: Bill Dress, Ray Duncan
Secretary (appt'd. by chair): Martin Tracy
International Representative: Larry Forsley
Vocabulary Representative (appointed by the chair): Ted Dickens
Documentation Editor (appointed by the chair): Ron Braithwaite

Applications for these positions may be accepted until August 31st. Each requires a letter of intent and qualification, along with a letter from your employer indicating that they understand the amount of time needed for taking on these assignments.

Greg Bailey and Don Colburn then proposed that the technical subcommittee agree to mark up the standards document, indicating areas to be deleted, modified, added, and areas of deviation from accepted practice. This was agreed 13-3.

The group then agreed 9-4 to hold the next meeting on November 11-12, 1987 in Southern California at FORTH, Inc.

Elizabeth named acting chairs for each of the subcommittees: Ted Dickens, documentation; Gary Betts, logistics; Guy Kelly, research; Greg Bailey, technical.

I passed out documentation and gave a brief description of the ANSForth bulletin board on MCI Mail.

There was then a review of all action items and the TC adjourned. This was immediately followed by the convening of the Technical Subcommittee (TSC).

Martin Tracy volunteered to serve as acting TSC secretary. There was a lengthy discussion on the proper name for this subcommittee (I’m using TSC in these notes, but that may not be accurate) and its voting membership requirements. At issue was whether it was a formal subcommittee which would carry additional documentation requirements. As I recall, no conclusion was reached on this. We did get the impression that in order to be a voting member of the TSC, you had to be a voting member of the TC.

The TSC then drew together a list of goals: identify a kernel of highly compatible words, decide a strategy for layering and extensions, amass information on the TC desires and needs, and define a mechanism for handling proposals.

I ran out of steam around this point and stopped taking notes, but most of the remaining discussion was on the proposal process, voting membership, and plans for the next meeting.

Commentary

While the preceding describes events to the best of my recollection and note-taking abilities, I thought I’d add a few opinions and observations of my own:

First, I think this effort is off to a great start. The membership includes an excellent and reasonably well-balanced group of vendors, users, and other interested folks. I believe most of the early objections to this effort have been resolved by the make-up of the TC. Additionally, there was a clearly cooperative spirit among the attendees.

It was very clear to me that the TC was determined to pursue an open organization. Several discussions were concerned with how to publicize our procedures and encourage participation. This, along with CBEMA’s requirements for “due process” will, I think, result in an excellent document.

Unfortunately, it seems there is still an IEEE cloud hanging over this effort. I thought a compromise had been reached, but apparently there are still a few people pursuing the IEEE Forth alternative. We’ll have to wait and see what happens.

Chuck Moore, in spite of his avowed opposition to a Forth standard, was extremely cooperative. My impression is that he was mainly concerned with having wide participation and not shutting off the possibility of new Forth development. (He also mentioned that he was working on a new Forth compiler.)

Don Colburn seemed to feel that we could put out a draft document much earlier than planned and was surprised at the idea that there would be any difficulty in reaching a consensus.

I felt that Elizabeth did an excellent job of chairing the meeting, but suppose she was wise in not volunteering as the permanent Chair. This way, we avoid even the appearance of a FORTH, Inc.-dominated effort.

I have a couple of concerns about the course of this project. Most difficult for me to reconcile is the notion of a standard documenting common usage among the major Forth implementations. In some cases, this may cause a reversion of some language features back to the way they were before the 83 standard. For example, FORTH, Inc. never changed its definition of leave to correspond with the 83-standard; i.e., it does not immediately leave the loop. One could, therefore, argue that the 83 standard leave is not in common usage. Thus, it seems to me that the ANS standard might either leave its effect undefined or else omit it entirely. Worse, I think, would be to revert its meaning back to the 79 standard.

My other main concern is with the minimalist approach. I guess it’s the only sensible way of getting this out in a reasonable amount of time, but I worry that most proposals will simply be put aside with the note that they’re outside the documented scope of the ANS Forth effort.

On the positive side, I think this group has enough talent and dedication to complete a superb standard in a reasonable amount of time. I offer my personal thanks to everyone involved for providing two days of stimulating discussion. A special tip of my Forth beanie goes to Elizabeth Rather and Martin Tracy for doing the bulk of the work in pulling this activity together.

Other Notes of Interest

Don and Chris Colburn were kind enough to invite everyone over to their house Monday evening for pizza and pool (swimming, that is). It was very pleasant and provided the opportunity for people to get to know each other a bit better. Don took us down for a tour of his workshop; it looked like a Mac farm. Don demonstrated the Mac II running several animated graphics tasks under MacForth in separate windows. Very nice.

I was very happy to have Martin Tracy and Guy Kelly stay at my house, but regret the short time available for Forth talk. Guy demonstrated his new, implementation-independent Forth editor. It seemed very powerful — it could work with screen files or native blocks. In addition, he provided three ways for moving stuff around — cut and paste, a line stack (push and pop a line at a time), and a "barrel" (push stuff into the barrel, and select from it in any order). I believe Guy will be offering this for sale.
MCI MAIL'S ANS FORTH BBS

MCI Telecommunications is sponsoring a bulletin board on MCI Mail in support of the ANSI Forth standards activity known as X3J14. This board will contain agendas, proposals, minutes of meetings, and related information. If you are interested in the development of the ANS Forth standard, this is the place to see what’s going on.

ANS Forth Bulletin Board

ANSForth is the main heading for several types of message areas. From the main "Command:" prompt, type VIEW ANSFORTH to see all currently active areas.

The following are currently available:
1. General: general information on X3J14; membership, documentation, officers, etc.
2. Agenda: agendas for X3J14 meetings.
3. Minutes: minutes of previous meetings.
5. Comments: comments on active proposals.
6. Misc.: uncategorized messages.

In general, bulletin board messages will only remain available for 90 days (this is an MCI Mail constraint). Archived versions of older messages will be available for downloading from the East Coast Forth Board.

Viewing the Bulletin Board

To access the ANSForth Bulletin Board type
VIEW ANSFORTH <subarea-name>

For example, type
VIEW ANSFORTH AGENDA

to see the upcoming agenda.

If you don’t know the particular subarea’s name, type at least ANSFORTH, then choose the subarea you want from the list of matching names. You will then see a "View:" prompt; type one of the following commands:

SCAN To display a scan table of items on the board, showing the date posted, subject, and the size of the item.

READ To display all items or those selected by scan number, with page breaks and a pause between items.

PRINT To display all items or those selected by scan number, with no page breaks or pauses between items.

LEAVE To exit the board and return to mail mode.

EXIT To log off MCI Mail.

Posting Messages on the Board

Unlike many other bulletin boards, MCI Mail only allows the bulletin board owner (me) to post messages. Therefore, in order to get a message posted for public view, you must send it to me to be forwarded.

The best way to send me a message is via MCI Mail. I check this mailbox daily, and forward messages to the bulletin board with just a few keystrokes. To send me a message on MCI Mail, enter the CREATE command and at the “TO:” prompt enter "Gerald A. Shifrin". Alternatively, you may address a message to my MCI Mail id: 299-4103.

The second best way to get a message posted is to leave it on the East Coast Forth Board at 703-442-8695, addressed to SYSOP with instructions for it to be posted on the ANS Forth Bulletin Board. Similarly, you may upload a file (also with appropriate instructions) to me. This will get your message posted within a day or two.

If you don’t telecommunicate but want to get a message posted, send me a floppy disk in IBM PC format containing the file(s) to be posted. Mail this to: Jerry Shifrin, 6212 Loch Raven Dr., McLean, VA 22101.

I’m sorry, but I won’t be able to return your disk unless you include a self-addressed, stamped mailer. Please do not send written material to be posted. I don’t have a document scanner and I’m not a great typist.

Prices

The annual mailbox fee is $18, payable upon registration and billed on the anniversary date of service. This allows you to read messages sent to you. There are additional charges for sending messages and access fees; these are described below. The initial $18 registration fee will be credited against any charges you have in the first two months.

It costs $.45 to send an instant message of up to 500 characters; $1 for up to 7500 characters, and another $1 for each subsequent 7500 characters. There is no connect-time charge if you call through a local MCI Mail number. There is a $.05/minute charge if you access MCI Mail via Tymnet.

—JS

[Editor's note: You may find the standards-related news and discussion on FIG’s GEnie conference (see announcement elsewhere), and on the East Coast Forth Board, timely and complete enough for most purposes.]
When I began learning Forth, I was particularly excited about finally being able to operate in high resolution on the TI without having to wrestle with assembly language. But I immediately ran into a snag. The transcendentials (sin, cos, etc.) compete with the bit-map color table for work space! No matter that I wasn't using any color commands — a messy screen resulted. How was I to draw a simple circle without sines and cosines? I know that I'm not the first to tackle this problem, and probably not the first to solve it, but here goes.

My first effort involved the use of a sin table. It worked. Then I saw a copy of FD IV/1 with the trig derivation screen by J. Bumgarner. With some slight modifications and a couple of additions, it became a part of my program. A reduction of memory overhead, and one less screen resulted. Some major changes in the graphics commands followed and GRIDPLOT was born.

The accompanying screens are reasonably well documented, so I won't discuss each word in detail. HUE sets the drawing color against a black background. XSET and YSET add your X and Y positions to XC and YC (constants describing the center of the screen) to effect X,Y plotting. CLEAN erases the bit-mapped screen, accounting for the current mode, GRAPHICS2, SPLIT, or SPLIT2. ?BREAK checks for FUNCTION CLEAR, allowing you to halt a graphic in progress.

A more descriptive discussion of CIRCLE will help in understanding the graphic words. The sin of the angle of a rotating radius at each step gives the X position. In a like manner, the cos produces the Y position. These successive X,Y coordinates are plotted, forming the perimeter of the circle. XSET and YSET shift these coordinates to the correct quadrant of the X,Y grid. (See Figure One.)

A complete understanding of TLINE is necessary, both because of the resident word CLINE, and because how it works is not immediately evident. CLINE writes the tiny characters to the bit-mapped screen. CLINE expects to find on the stack the address of the text to write, the character count, and the line number to write to. The line numbers are rows zero to 23, as if in the 32-column mode. CLINE always begins writing at column zero. The only way to move the text over is to pad it with 32s preceding the first character. That is a major flaw when you want to write on a screen containing graphics. Those blanks will erase anything in their way.

TLINE is a redefinition of CLINE. I didn't redefine CLINE itself because CLINE is still useful, and requires less effort on our part to use when it is appropriate.

Both TLINE and CLINE call SMASH, a resident Coded word which formats the tiny characters and puts them into a line buffer, LB. SMASH leaves on the stack the information expected by VMBW (Video Multiple Byte Write), the LB address, the VDP address to write to (column and row, to us), and the character count.

In TLINE, the VDP address left by SMASH is DROPed and replaced by the VDP address that is the column and row we want to write to. That address is computed by TPUT by adding the necessary number of addresses to skip, to the beginning of the bit-map pattern descriptor table — which starts at hex 2000 (8192), and is stored in the variable PUT. The text is stored in TBUFF, beginning at TBUFF+1. The count is stored at TBUFF (i.e., cnt TBUFF C!).

Use the keyboard method (TGET) to find just where you want to print your text, then edit the necessary commands onto your Forth screen. See the notes on screen 29 for examples of how to do it. As a reminder, TGET requires two returns.

Screen 31 contains a demonstration which should help in getting started. Screen 30 contains two formulas that draw some very satisfying graphics. Writing HYPO and EPI will be valuable practice. Then add the SCREEN_DUMP from FD VI/6 and you will have a toy well worth playing with!
SCh
I?
LRIDPLOT
[ver 2.0]
Loadino screen. GT Apr 85
BASE-:)R
DECIHAL
TEXT CLS 0 2 GOTOXY .
GRIDPLOT ver 2.0* CR
EMPTY-BUFFERS
C.A.F.I.S.* CR CR
. 20 - sin & cos functions* 20 LOAD CR
. 21 - variables and utilities* 21 LOAD CR
. 22 - utilities continued* 22 LOAD CR
. 23 - circle command* 23 LOAD CR
. 24 - rule & point commands* 24 LOAD CR
. 25 - sookes command* 25 LOAD CR
. 26 - polygon command* 26 LOAD CR
. 27 - box command* 27 LOAD CR
. 28 - tline (bit-map print)* 28 LOAD CR
15 R->BASE ." Command?" QUIT
SCR #20
0 ( TRIG by derivation[MOD]: 4th DIMENSIONS 4/1, J.Bumgarner )
1 BASE->R DECIMAL
2 10000 CONSTANT 10K ( scale factor )
3 : LF SWAP / MINUS 10K / 10K + ; ( repetitious formula )
4 : ?MIRROR DUP 90 > IF 100 SWAP - THEN ;
5 : REDUCE 360 MOD DUP 0 IF 360 + THEN DUP 180 < IF ?MIRROR
6 ELSE 180 - ?MIRROR MINUS THEN ; ( establish sector )
7 : SIN REDUCE 17453 100 / ( convert degrees to radians )
8 : DUP DUP 10K / >R 10K 72 R LF 42 R LF 20 R LF 6 R LF
9 10K / R> DROP ; ( calculate sin )
10 : COS 360 MOD 90 SWAP - SIN ; ( COS(X) = SIN(X+90) )
11 : SINX ( n angle -- n*sin[angle] ) SIN M* 10K M/ SWAP
12 4999 > IF 1+ THEN ; ( rounded up if rem = ) .5 )
13 : COSX 360 MOD 90 SWAP - SINX ; ( plotable X,Y positions )
14 R->BASE :S
15 Modified by GT, Apr 85
SCR #21
0 ( GRIDPLOT COMMANDS, GT Apr 85 )
1 BASE->R HEX
2 ( Variables, Constants )
3 0 CONSTANT XC 60 CONSTANT YC ( center of screen constants )
4 0 VARIABLE XP 0 VARIABLE YP ( x and y positions )
5 0 VARIABLE RAD ( radius value ) 0 VARIABLE INC ( increment )
6 0 VARIABLE XP' 0 VARIABLE YP' ( TO equivalent of XP, YP )
7 0 VARIABLE WID 0 VARIABLE DEP ( box variables )
8 : HUE ( c -- ; 1 cyan, 2 red, 3 yellow, 4 magenta, 5 green )
9 CASE 1 OF 71 ENDOF 2 OF 61 ENDOF
10 3 OF A1 ENDOF 4 OF D1 ENDOF
11 5 OF C1 ENDOF 6 OF F1 ENDOF ( white )
12 ENDCASE DCOLOR !; ( default is white )
13 : XSET XC XP @ ++ ; ( X and Y offsets for plotting on )
14 : YSET YC YP @ ++ ; ( X,Y grid )
15 R->BASE

Yes, 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)! With over 1500 functions you are almost done before you start.

WELCOME TO HS/FORTH, where megabyte programs compile at 10,000 lines per minute, and execute faster than ones built in 64k limited systems. Then use AUTOOPT to reach within a few percent of full assembler performance -- not a native code compiler linking only simple code primitives, but a full recursive descent optimizer with almost all of HS/FORTH as a useable resource. Both optimizer and assembler can create independent programs as well as code primitives. The metacompiler creates threaded systems from a few hundred bytes to as large as required, and can produce ANY threading scheme. And the entire system can be saved, sealed, or turned off for distribution either on disk or in ROM (with or without BIOS).

HS/FORTH is a first class application development and implementation system. You can exploit all of DOS (commands, functions, even shelved programs) and link to GIB and LIB files meant for other languages interactively.

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, and greatly enhance both time slice and round robin multitasking utilities. Math facilities include both software and hardware floating point plus an 18 digit integer (financial) extension and fast arrays for all data types. Hardware floating point covers the full range of trig, hyper and transcendental math including complex.

Undeniably the most flexible & complete Forth system available, at any price, with no expensive extras to buy later. Compiles 79 & 83 standard programs. Distribute metacomplied tools, or turnkeyed applications royalty free.

HS/FORTH (complete system): $39.5
HS/FORTH: Tutorial & Ref (500 pp) $ 59
Forth Text & Reference (500 pp) $ 22
HS/FORTH Glossary $ 10
GIGAFORTH Option (Beta release) $425.
(Native Mode from SOFTMLL, INC.)
SCR #22
0 (GRIDPLOT COMMANDS, 6T Apr 85 ) BASE->R DECIMAL
1 ( Utilities, cont’d )
2 : CLEAN ( -- erase bit-map screen )
3 PAD 64 32 FILL ( ensure only blanks )
4 VDPMDE @ CASE ( get bit-map mode )
5 (graphics2) 4 OF 24 0 ENDOF
6 (split)  5 OF 16 0 ENDOF
7 (split2) 6 OF 24 4 ENDOF
8 ENDCASE
9 DO PAD 64 I CLINE LOOP ; ( erase each line )
10
11  ?BREAK ( ;p check for break [fctn clr] if in graphics2 then )
12  ( go to split [to make keyboard text visible] )
13  ?TERMINAL DUP DUP IF VDPMDE @ 4 = IF SPLIT THEN IF CLEAN
14  QUIT THEN THEN ;
15 R->BASE

SCR #23
0 (CIRCLE; GRIDPLOT 6T Apr 85 ) BASE->R DECIMAL
1 : CIRC ( [variables] circle -- )
2 361 I DO RAD @ I SIND XSET ( calculate column )
3 RAD @ I COS@ YSET ( calculate row )
4 DOT ?BREAK ( plot row, col; fctn clr? )
5 INC @ iLOOP ; ( plot only every INC deg )
6 : CIRCLE ( xp yp rad -- ;p xp=0, yp=0 is center of screen )
7 DUP 35 > IF 1 ELSE DUP 15 < IF 3 ELSE 2 ( set inc )
8 THEN THEN INC !
9 RAD ! YP ! XP ! CIRC ;
10 R->BASE ;S There are some INC’s and DO limits which in
11 combination produce smoother circles at some radii. Limited
12 resolution of 256 x 192 tends to make ragged circles.
13 X,Y quadrant: upper lt lower lt upper rt lower rt center
14 XP! neg neg pos pos 0
15 YP! neg pos neg pos 0

SCR #24
0 (RULE, POINT; GRIDPLOT 6T Apr 85 ) BASE->R
1 : RULE ( xp yp xp’ yp’ -- ;p draw line using X,Y coord. )
2 >R >R >R X; + R> YC + ( col, row from )
3 R> IC + R> YC + ( col, row to )
4 LINE ; ( draw line )
5
6 : POINT ( xp yp -- ;p plot dot using X,Y coord. )
7 >R X; + R> YC + DOT ; ( 0 0 POINT is scrn center )
8
9 R->BASE ;S
10
11
12 RULE draws a line from/to X,Y coordinates. POINT plots a dot
13 at X,Y coordinate. LINE and DOT commands may still be used as
14 usual to draw a line from col,row to col,row, and dot to plot a
15 pixel on at col,row.

 NOW FOR IBM PC, XT, AT, PS2 AND TRS-80 MODELS 1, 3, 4, 4P

The Gifted Computer

1. Buy MMSFORTH before year’s end, to let your computer work harder and faster.
2. Then MMS will reward it (and you) with the MMSFORTH GAMES DISK, a $39.95 value which we’ll add on at no additional charge!

MMSFORTH is the unusually smooth and complete Forth system with the great support. Many programmers report that in ten times greater productivity with this outstanding system, and MMS provides advanced applications programs in Forth for use by beginners and for custom modifications. Unlike many Fortes on the market, MMSFORTH gives you a rich set of the instructions, editing and debugging tools that professional programmers want. The licensed user gets continuing, free phone tips and a MMSFORTH Newsletter is available.

The MMSFORTH GAMES DISK includes arcade games (BREAKFORTH, CRASHFORTH and, for TRS-80, FREEWAY), board games (OTHELLO and TIC-TAC-TOE, and a top-notch CRYPTOQUOTE HELPER with a data file of coded messages and the ability to encode your own. All of these come with Forth source code, for a valuable and enjoyable demonstration of Forth programming techniques.

Hurry, and the GAMES DISK will be our free gift to you. Our brochure is free, too, and our knowledgeable staff is ready to answer your questions. Write. Beller yet, call 617/853-6136.

GREAT FORTH:
MMSFORTH V2.4.......$179.95
The one you’ve read about in FORTH:A TEXT & REFERENCE. Available for IBM PC/XT/AT/PS2 etc., and TRS-80 M.1, 3 and 4

GREAT MMSFORTH OPTIONS:
FORTHWRITE .................$99.95
FORTHCOM ..................49.95
DATABASEHANDLER ............59.95
DATABASEHANDLER-PLUS* ....99.95
EXPERT-2 ....................69.95
UTILITIES ..................49.95

*Single-computer, single-user prices; corporate site licenses from $1,000 additional. 3½” format, add $5/disk; Tandy 1000, add $20. Add S/H, plus 5% tax on Mass. orders. DH+ not available for TRS-80s.

GREAT FORTH SUPPORT:
Free user tips, MMSFORTH Newsletter, consulting on hardware selection, staff training, and programming assignments large or small.

GREAT FORTH BOOKS:
FORTH: A TEXT & REF.........$219.95
THINKING FORTH ...........16.95
Many others in stock.

MILLER MICROCOMPUTER SERVICES
61 Lake Shore Road, Natick, MA 01760
(617/853-6136, 9 am - 3 pm)
SCR #25
0 ( SPOKES: GRIDPLOT GT Apr 85 ) BASE->R DECIMAL
1
2 : SPO ( [variables] spokes -- )
3 361 0 DO RAD @ I SINV YSET ( X end position )
4 RAD @ I COS@ YSET ( Y end position )
5 XP @ XC + YP @ YC + ( origin of spokes )
6 LINE ?BREAK ( draw origin to X,Y )
7 XP @ +LOOP ; ( repeat rep times )
8 : SPOKES ( xp yp len rep -- ; p plot rep spokes len from xp,yp )
9 360 SWAP / XP' ! ( XP' used to store repititions )
10 RAD ! ( store len )
11 YP ! XP ! ( store origin of spokes )
12 SPO ;
13 R->BASE ;S
14 SPOKES plots spokes (as on a wheel) given a center, length of
15 spokes and number of spokes.

SCR #26
0 ( POLY; GRIDPLOT, GT Apr 85 ) BASE->R DECIMAL
1 : POL ( [variables] poly -- ; p draw polygon )
2 361 INC @ DO RAD @ I SINV YSET ( X end position from )
3 RAD @ I COS@ YSET ( Y end position from )
4 RAD @ I XP' + SINV YSET ( X position to )
5 RAD @ I XP' + COS@ YSET ( Y position to )
6 LINE XP' @ ( draw side )
7 +LOOP ; ( repeat 360/#sides times )
8
9 : POLY ( xp yp rad #side -- ; p draw polygon )
10 360 SWAP / DUP XP' ! INC ! ( store #side and inc )
11 RAD ! ( store radius )
12 YP ! XP ! ( store center position of polygon )
13 POL ;
14 R->BASE ;S POLY plots a polygon by connecting the ends
15 of spokes, but without drawing the spokes.

SCR #27
0 ( BOX PLOT, GT Apr 85 ) BASE->R DECIMAL
1 : RESET XP' @ XP ! YP' @ YP ! ; ( reset var to init value )
2 : XP+ XP @ WID # + XP ! 0 ISET ; ( calulateate corners )
3 : YP+ YP @ DEP # + YP ' 0 ISET ;
4 : BX ( [VARIABLES] box -- ) XP @ YP' ! XP @ XP' ! ( save var's )
5 ( col,row fm ) ( col,row to )
6 0 ISET 0 YSET XP+ 0 YSET LINE RESET ( draw line & )
7 0 ISET XP+ 0 YSET LINE RESET ( restore var's )
8 0 ISET 0 YSET 0 ISET XP+ LINE RESET
9 XP+ 0 YSET OVER XP+ LINE ;
10
11 : BOX ( xp yp wid dep -- ; p ul corner, wid accross, dep down )
12 DEP ! WID ! YP ! XP ! BX ;
13 R->BASE
14
15
FEATURES

- FORTH-79 Standard Sub-Set
- Access to 8031 features
- Supports FORTH and machine code interrupt handlers
- System timekeeping maintains time and date with leap year correction
- Supports ROM-based self-starting applications

COST

130 page manual — $30.00
8K EPROM with manual — $100.00

Postage paid in North America. Inquire for license or quantity pricing.

Bryte Computers, Inc.
P.O. Box 46, Augusta, ME 04330
(207) 547-3218

SCR #28

0 TLINE: GRIDPLOT 6T Apr 85 ) BASE->R DECIMAL
1
2 0 VARIABLE TBUF 34 ALLLOT ( buf for 64 char of TLINE text )
3 ( PLUS char count, )
4 0 VARIABLE PUT ( PUT holds the VDP address to write to, the )
5 ( address is converted by TPUT from col, row, )
6 7 : TCNT TBUF 1+ 64 0 DO DUP I + C8 0= IF I LEAVE THEN
7 ( count from 1st char of text to first null, put the )
8 ) ( count on the stack and LEAVE the loop, )
9 10 LOOP SWAP 1- C! ; ( store char count at TBUF )
11 12 : TPUT 256 # SWAP 4 $ + B192 + PUT ; ( col, row -- )
13 ( calculate VDPaddr to begin writing at, store in PUT, )
14 R->BASE -->
15

SCR #29

0 TLINE: GRIDPLOT 6T Apr 85 ) BASE->R DECIMAL
1 : TGET TBUF 1+ 64 EXPECT TCNT ; ( -- <enter> text <enter> )
2 ( store text beginning at addr TBUF plus 1, )
3 4 : TLINE LB 100 ERASE SMASH >R DROP ( drop addr from dummy arg)
5 ( and replace it with) PUT @ R) VMWB ; ( write on screen )
6 ( LB is a resident linebuffer used by SMASH, )
7 ( SMASH requires a line #, thus the dummy argument, )
8 : WRITE TBUF COUNT 1- 0 ( dummy argument) TLINE ; ( -- )
9 10 R->BASE ;S To print from screen use the string-store word,
11 !" and put the text at TBUF 1+, then put count at TBUFF, give
12 the col, row to TPUT; TBUF 1+ '" TEXT" cnt TBUF C! col row TPUT
13 Then execute WRITE. To print from keyboard use TGET,TPUT,WRITE.
14 ALWAYS MAKE CHAR COUNT EVEN, PAD WITH A BLANK IF NECESSARY; AND
15 START ON AN EVEN NUMBERED COLUMN. ROWS ARE 0 THRU 23.

SCR #30

0 ;S GRIDPLOT - Some interesting formulas for graphics. The
1 formulas are given in algebraic form. It is left to you to
2 put them into post-fix notation.
3 Epicycloid: X = (A+B)#COS(theta) - A#COS(A+B/A#theta)
4 Y = (A+B)#SIN(theta) - A#SIN(A+B/A#theta)
5 Hypocycloid: X = (A-B)#COS(theta) + B#COS(A-B/B#theta)
6 Y = (A-B)#SIN(theta) - B#SIN(A-B/B#theta)
7
8 Theta is the same rotating angle used to describe a circle.
9 A and B are the respective radii of a large and small circle,
10 one rolling around the other. Imagine the one rolling has a
11 pencil attached and that the pencil is tracing a line. This is
12 the graphic produced by the above formulas.
13 Epicycloid=small wheel rolling around the outside of the larger.
14 Hypocycloid=small rolling around the inside of the larger.
15 6T Apr 85
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, //gs version late Fall) and the Apple® ////. It is 100% ProDOS® and SOS ® supported. Go FORTH code is intercompatible 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 support 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 (Applesoft-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
3201 Murchison Way, Carmichael, California 95608
(916) 485-6525

Apple //e, //c, //gs and ///, ProDOS and SOS are registered trademarks of Apple Computer, Inc. No affiliation with Pair Software
Screen # 6
( Question 1 options
jbb 12:16 10/12/86 )
: .Q@OPTIONS GREEN PG 0 3 TAB
" Diagram Elements:" CR CR GRAY PG
" 1. Centrifugal Force" CR
" 2. Force Normal" CR
" 3. Force Parallel" CR
" 4. Weight" CR
" 5. Weight Normal" CR
" 6. Weight Parallel" CR
" 7. \* theta \* (Theta)" CR
" 8. Center of Gravity" CR
" 9. Force Parallel" CR
" Superelevation Rate (e) ENTRY-W ;

: MATCH SETUP SLOPE CG-MARK CF-DiAG W-DiAG TAG \*ENTRY-W ;

Screen # 7
( Question 2 options
jbb 12:16 10/12/86 )
: .Q@OPTIONS GREEN PG 0 3 TAB
" Diagram Formulae:" CR CR GRAY PG
" 1. W = V^2/GR CR
" 2. W = 253 EMIT \* sin \* theta \* / GR \* CR
" 3. W = 253 EMIT \* cos \* theta \* / GR \* CR
" 4. W = GR CR
" 5. W = cos \* theta CR
" 6. W = sin \* theta CR
" 7. arctan e" CR
" 8. Center of Gravity" CR
" 9. rise per foot " ENTRY-W ;

: MATCH2 MATCH .Q@OPTIONS ;

Screen # 8
( Answer processing
jbb 12:15 10/12/86 )
VARIABLE ANS
( holds correct answer)
: QUESTION ( - ) "; What is letter \" ;
: GET-ANS ( a - ) \$IN ;
: ANSWER ( a - ) ANS ;
: CHECK-ANS ( n - ) ANS @ - IF \" WRONG \" ELSE \" RIGHT \" THEN ;
: SHOW-ANS ( a - ) CR "; The correct answer is \" ANS \" ;
: WAIT ( - ) "; Press a key to continue." KEY DROP ;
: EVALUATE ( - ) GET-ANS CR CHECK-ANS SHOW-ANS CR CR WAIT ;

Note: these are "down & dirty" quick demo versions of answer processing routines. There are no keyboard error traps or variations in feedback. They serve only to illustrate the functions for the quiz demo.

Screen # 9
( Quiz
jbb 12:13 10/12/86 )
: FINDCHAR ( i - a ) 97 + A$ \$ IN \$ ;

( find character's position in A$)
: MATCHANSWER ( a - ) 2* ANSWERS + \$ ANSWER ;

( match position to answer in ANSWERS)
: GIVEQUIZ ( - ) 11 0 DO 1 FINDCHAR MATCHANSWER ENTRY-W QUESTION 1 97 + EMIT \" \" \* CR EVALUATE LOOP ;

: Q1 MATCH1 GIVEQUIZ ;
: Q2 MATCH2 GIVEQUIZ ;
: Q3 Q1 Q2 ;

(Continued from page 10.)

8

01MARB&AHIT

Test answer by comparing to reference data. If correct response, increment question count. If incorrect, show correct answer but don't increment count. Congratulations for good work in completing conversion quiz.

Obtain a number and display it to student. Keep a copy for subsequent evaluation of response.

Present a number in output base. Accept answer in input base.

If correct c'=c+1, else c'=c.

Define a general quiz administrator to present five numbers for conversion.

Define a quiz for decimal-to-binary conversion by setting data output base to 10 and data input base to 2.

A binary-to-decimal quiz.

9

01MARB&AHIT

A hexadecimal-to-binary quiz.

Binary-to-hexadecimal.

How about base 3-to-base 5?

Binary-to-octal.

Octal-to-binary.

Decimal-to-hexadecimal.

Hexadecimal-to-decimal.
and that this is usually known after comparing just the first character. With the link field first, the search routine can access the link immediately. Otherwise (except on constant-length name field systems), it has to AND the length byte with 31, then ADD this to the previous link field to determine the location of the current link field.

3. 6502 fig-FORTH on a Commodore 64, 8080 fig-FORTH on an Exidy Sorcerer, and an ancient, non-standard PDP-11 version of Forth, so old EMIT was still called SPIT.

4. In my hacking manner, I only recently divided SEE into the components ISEE and see — it had been just one big word. The wisdom of the breakdown became apparent after reading Thinking Forth, and after noting that every so often I needed to SEE a word starting somewhere after the header. I may yet begin to plan.

Rich Franzen works as an image processor/programmer for Satellite Exploration Consultants, Inc., where he uses AIMS, a very powerful image processing system that is written in an old, PDP-11 version of 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, 8006, 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.
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) 227-8717

• CALIFORNIA
  Los Angeles Chapter
  4th Sat., 10 a.m.
  Hawthorne Public Library
  12700 S. Crensworth Ave.
  Phillip Watson (213) 649-1428
  Monterey/Salinas Chapter
  Bud Devins (408) 633-3253
  Orange County Chapter
  4th Wed., 7 p.m.
  Fullerton Savings
  Huntington Beach
  Noshir Jessang (714) 842-3032
  San Diego Chapter
  Thursdays, 12 noon
  Guy Kelly (619) 450-0553
  Sacramento Chapter
  4th Wed., 7 p.m.
  1798-59th St., Room A
  Tom Ghormley (916) 444-7775
  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.
  Steven Sams (303) 477-5955

• 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 Forberg (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 Sizzle, Doraville
  Nick Hennigent (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 Kusilek (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.
  IP Univ. Campus, B711 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

• KANSAS
  Wichita Chapter (FIGPAC)
  3rd Wed., 7 p.m.
  Wilbor E. Walker Co.,
  532 Market
  American Science
  (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) 588-9522

• 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.
  Thombill 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
  Israel 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 Granger (513) 257-6984

• 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
• 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.
800 Oak Ridge Turnpike,
Richard Secrist (615) 483-7242

• TEXAS
Austin Chapter
Contact Matt Lawrence
P.O. Box 190409
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
Perlmans 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 VanSuykel (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.
Jock Shprenta (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 Tregear
10 Binda Rd., Yowie Bay
02/524-7490

• BELGIUM
Belgium Chapter
4th Wed., 20:00h
Contact Luk Van Loock
Larikssteeg 20
2120 Schoten
03/658-6343
Southern Belgium Chapter
Contact Jean-Marc Berinchamps
Rue N. Monnom, 2
B-6290 Nalinnes
011/213838

• 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. Solntsef (416) 525-9140
ext. 3
Toronto Chapter
Contact John Clark Smith
P.O. Box 230, Station H
Toronto, ON M4C 5Z2
Vancouver Chapter
Don Vanderweele (604) 941-4073

• COLOMBIA
Colombia Chapter
Contact Luis Javier Parra B.
Aplo. Aerico 100394
Bogota 214-0345

• DENMARK
Forth Interesse Groupe
Denmark
Erik Oestergaard, 1-52094

• 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
(16-6144.03.06
FIG des Alpes Chapter
Annely
Georges Seibert, 50 57 0280

• GERMANY
Hamburg FIG Chapter
4th Sat., 1500h
Contact Horst-Gunter Lunsche
Schanzenstrasse 27
2000 Hamburg 6

• HOLLAND
Holland Chapter
Contact Adriaan van Roosmalen
Heusden Houtestraat 134
4817 We Breda
31 76 713014

• IRELAND
Irish Chapter
Contact Hugh Dobbs
Newton School
Waterford
051/77577 or 051/74124

• ITALY
FIG Italia
Contact Marco Tausel
Via Gerolamo Fomi 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 Faerlas, 47-518-7784

• REPUBLIC OF CHINA
(R.O.C.)
Contact Ching-Tang Tseng
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

• FIGGRAPH
Howard Pearlmutter
(408) 425-8700

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

Volume IX, Number 3 39
ANNOUNCING

FORTH INTEREST GROUP ROUNDTABLE

NOW AVAILABLE THROUGH

GEnie™
General Electric Network for Information Exchange

* OVER 400 DOWNLOADABLE FILES OF FORTH INFORMATION & CODE
* ON-LINE REAL TIME CONFERENCING
* E-MAIL CONTACT WITH OTHER FIG MEMBERS

SPECIAL SIGN-UP FOR FIG MEMBERS ONLY

$18.00
Includes GEnie Manual plus 3 FREE Hours on GEnie
Using Your Modem Call 1-800-638-8369, Type "HHH" (CR)
Following the U# prompt, type "XJM11849,GENIE" (CR)
See Page 17 For Details

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