

F O R T H

D I M E N S I O N S

■
MODULE MANAGEMENT

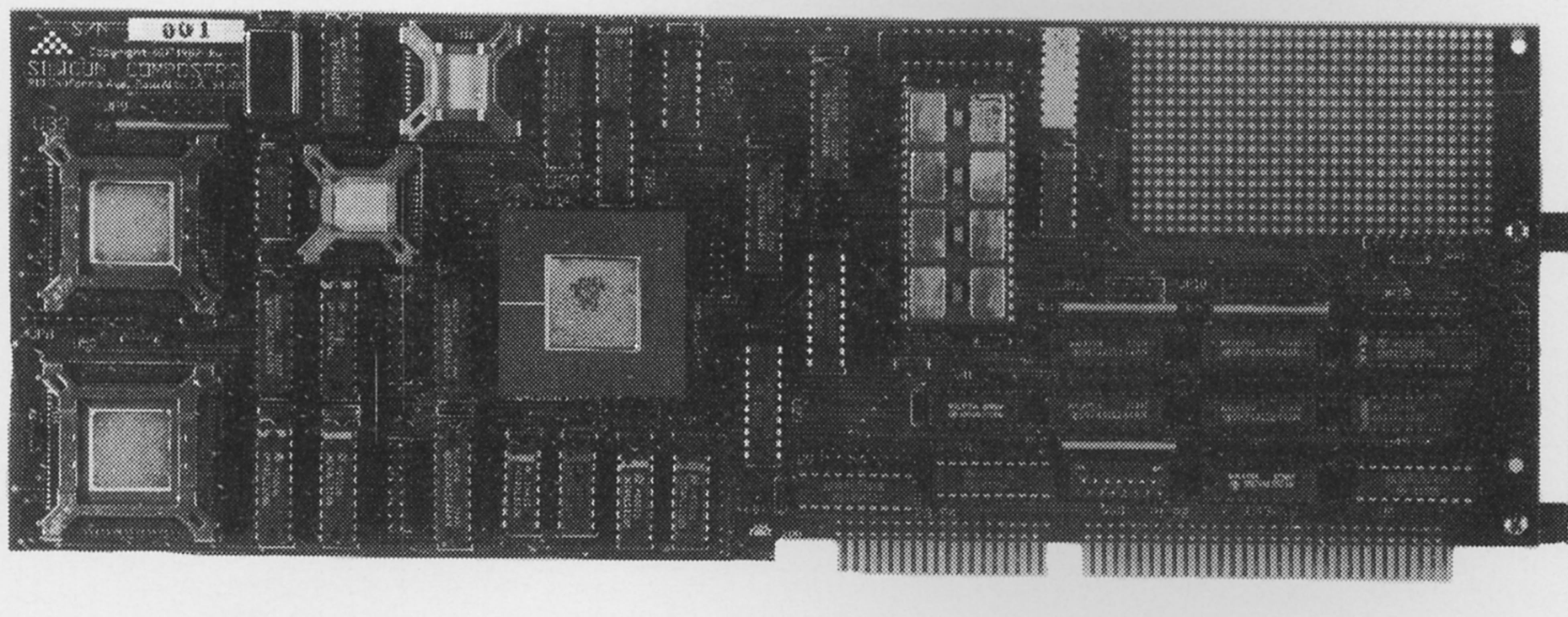
A 6502 ASSEMBLER

F83 FULL-SCREEN EDITOR

A FREE SPIRIT
■

ANOTHER STEP FORTH...

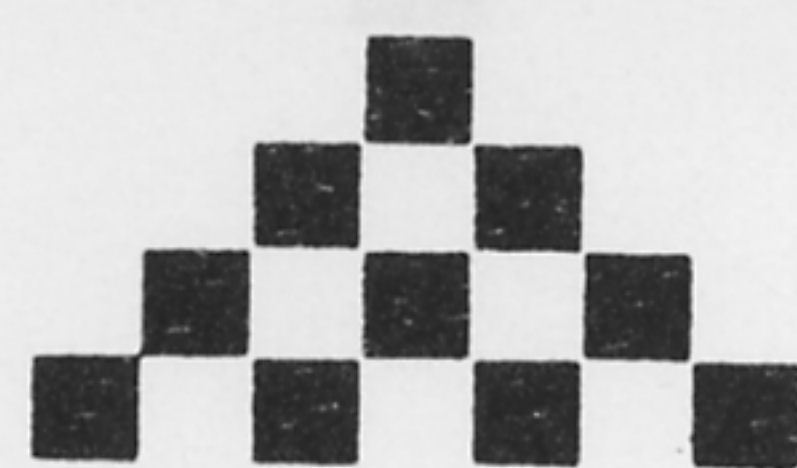
INTRODUCING THE AT/FORCE DEVELOPMENT SYSTEM



Using the FORCE Core Set from Harris Corporation

- AT-based Forth coprocessor
- 32K bytes main memory
- Expandable to 128 bytes
- Shared memory space with AT
- 2" × 3" prototyping area
- Forth System included
- Delivery: 2 weeks ARO
- Five chip Core Set includes:
 - FORCE Forth Core Engine,
 - one-cycle 16 × 16 multiplier,
 - two 64-word stack controllers,
 - 15-channel interrupt controller
- Five MIPS speed
- Price: \$4995

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



SILICON COMPOSERS

F O R T H

D I M E N S I O N S


A FREE SPIRIT: WHERE TO FROM HERE? • BY GLEN B. HAYDON

7

Each Forth user seems to have his own philosophy, religion, and brand of the language. Each certainly has his own expectations. This free spirit made Forth what it is and, at the same time, led to its lack of general acceptance. Programmers using other languages have never experienced such a free environment.

MODULE MANAGEMENT • BY ALAN T. FURMAN

9

 This module management system is a way of giving a symbolic name to a group of screens that contain generally reusable source code. That name is a Forth word that guarantees the presence in the dictionary of the code to be used by applications.

1987 FORTH NATIONAL CONVENTION • REVIEWED BY JERRY SHIFRIN

14

The Forth Interest Group (FIG) held its ninth annual convention on November 13-14 in San Jose, California. The theme was the "Evolution of Forth," eliciting much discussion about Forth's philosophical roots and its future.


ANS FORTH MEETING NOTES • BY JERRY SHIFRIN

16

The second meeting of the ANS Forth Technical Committee found only slow progress, with few usage questionnaires returned. However, areas of consensus and controversy were identified and matters of procedure were clarified. The stage may now be set for the real, productive work of the team.

A 6502 ASSEMBLER • BY CHESTER H. PAGE


19

 The only weakness I have encountered in Forth is the unavailability of primitive subroutines, called by JSR. This Forth 6502 assembler was designed not for long programs, but for assembling primitive words one by one. It requires only that the computer is 6502 based.

VECTORED EXECUTION & AN F83 FULL-SCREEN EDITOR

BY RICHARD E. HASKELL & ANDREW MCKEWAN

24

 Vectored execution is useful for directing flow of control. Different types of jump tables are often more convenient, and execute faster, than a corresponding CASE statement. One form of jump table will be illustrated by an F83 full-screen editor. (F83 and fig-FORTH)

PROFILES IN FORTH: JOHN D. HALL

31

Interviewer Mike Ham caught up with the Forth Interest Group director best known to most members as the official FIG Chapters Coordinator. John shares his insider's view of FIG, Forth, and the future.

Editorial

4

Letters

5

Advertisers Index

35

Rumor Stack

37

FIG Chapters

38

EDITORIAL

I'd like to welcome the newly elected, re-elected, and continuing members of the Forth Interest Group's Board of Directors. They all bring talent, energy, and experience to their positions, and are dedicated to furthering the causes of the Forth community. Your support and constructive input will empower them in the job they are doing for the rest of us.

There has been so much discussion recently about Forth's future and the possible directions FIG can take, that this issue emphasizes some of the issues and viewpoints. John Hall and Glen Haydon, in particular, touch on important areas of concern.

About Forth's future, may I say that the people who most fervently ask that question seem to be in businesses where Forth's relatively iconoclastic methods of accomplishing tough tasks make management edgy. But productive companies are quietly using Forth every day to write important, profitable code. They employ professional programmers and don't spend much time promoting Forth or worrying about its future. They are in business to get a job done, and are using a language that — with cultivation and experience — adapts entirely to their specific needs and practices.

If you find an ANSI Forth document in the future that doesn't include *your* practice, then maybe you were one of the 250 key people who didn't return a questionnaire to the Technical Committee, or maybe you buy your Forth from one of them... Jerry Shifrin's concluding notes about the last ANS Forth meeting are interesting and important. As difficult as it is to imagine a standards document that can codify common practice, how close it gets will depend on who gets involved.

Any proposal submitted to the commit-

tee is open to public comment, and all such comments must be addressed in committee meeting. So there is opportunity for wide participation, even without becoming an official member of a committee.

Opportunities will abound Down Under on May 19-20, 1988, when the first Australian Forth Symposium will be held at the NSW Institute of Technology. There will be a keynote presentation by Charles Moore (Forth's inventor), and the Novix Forth microprocessor family will be featured. The first day will include papers and demonstrations to show what can be done in Forth, and how quickly working applications can be developed. The second day will offer a choice of workshops, with instruction and hands-on experience. An exhibit will be running both days. If you are interested in a possible group travel rate from the United States, call the Forth Interest Group for information or see the ad in this issue.

This symposium has been initiated by a group of professionally based Forth users (from both industrial and academic organizations) who believe the language should be more widely known and used by professionals. The focus will be on Forth as a programming system for productivity, and papers are still welcome. We've heard for some time that Australia and New Zealand have some pretty interesting Forth activity, so this will be a great chance to learn how business is progressing in that part of the world and how their Forth professionals are planning for the future. (And 1988 is Australia's bi-centennial year, if you needed just one more reason to go....)

—Marlin Ouverson
Editor

Forth Dimensions

Published by the

Forth Interest Group

Volume IX, Number 5

January/February 1988

Editor

Marlin Ouverson

Advertising Manager

Kent Safford

Design and Production

Berglund Graphics

ISSN#0884-0822

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

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

Copyright © 1987 by Forth Interest Group, Inc. The material contained in this periodical (but not the code) is copy-righted by the individual authors of the articles and by Forth Interest Group, Inc., respectively. Any reproduction or use of this periodical as it is compiled or the articles, except reproductions for non-commercial purposes, without the written permission of Forth Interest Group, Inc. is a violation of the Copyright Laws. Any code bearing a copyright notice, however, can be used only with permission of the copyright holder.

About the Forth Interest Group

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

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

LETTERS

Local Variables

Dear Editor,

Local variables have been the subject of many earlier contributions, but I have not seen the following, simple implementation.

My version of Forth local variables makes use of ordinary Forth variables declared, and probably used, outside the colon definition. The only word to be used, following the name of the variable, is LOCAL. This must be in the beginning of a colon definition, and not inside control structures or some other kind of return-stack manipulation. The variable can then be used freely inside the colon definition, and will be restored to its original value on exit.

The code for LMI PC/FORTH+ is shown in Figure One. (For PC/FORTH, omit ADDR>S&O.)

Since the natural scope for a local variable in Forth is a colon definition, local variables can be managed on the return stack. LOCAL first saves the address and the value of the variable on the return stack, then arranges an exit through (LOCAL) by placing the address of (LOCAL) on the return stack. On exit from the colon definition, which was the local scope of the variable, (LOCAL) will then restore the old value of the variable from the return stack. There can be more than one local variable in a definition.

For a simple example, see Figure Two's rather foolish implementation of the factorial function N! with recursion and a local variable.

This is a simple and easy-to-use, high-level implementation of local variables. An

assembler-coded version would probably provide very fast local variables in Forth.

Yours,

Henning Hansen

116, Technical University of Denmark
2800 Lynby, Denmark

Don't Chip Off the Old Block

Dear Marlin,

I read, too, in *ForthDimensions* (VII/2), Mr. Ramer W. Streed's letter about changing source editing. Well, I must say that sometimes it's also difficult to me to get rid of screen numbers, lines, and shadows. (First I was using Super-Forth 64 on the little Commodore; now I own an IBM compatible, running a modified version of F83 — and I only got F83 two or three weeks ago!) Anyway, I don't agree with Mr. Streed completely...

Surely it can be useful, being able to read and compile a text, source file, especially if it is downloaded from a BBS, but I think we must keep screens. I mean, screens are part of Forth philosophy, of thinking Forth! It is BLOCK, SCR, BLK, BUFFER, LIST, and FLUSH that make Forth different from other languages, and therefore so fascinating!

I don't think screens are so awful. Especially with F83's shadows and glossary, it is possible to supply all the documentation needed, even for a large program to be understood. Abandoning screens for ASCII would *negate* Forth philosophy and make Forth similar to other, non-documenting languages.

I agree with Mr. Streed when he says

that, when we have developed a low-level word, we don't have to concern ourselves with what it has to do when called from within another one. But he means the opposite thing when he says that someone must keep track of line numbers, screens, and so on. Forth is beautiful, because when you write a word, you can immediately test it on the keyboard — unlike other, so-called structured languages that, in reality, isolate the programmer from his computer (see the hateful Pascal, for example, which doesn't let you examine a single part of your program without having to compile it all...).

I mean, when I've developed a low-level word and, having retested it, found it works on a general job, I don't have to concern myself further with where it is (anyway, I can VIEW it). About moving lines, I think it is better to avoid crowding a screen with a lot of words from the beginning, or the documentation will be unuseable. I thank both Mr. Pasquale and Mr. Wenrich for their work, but I encourage all Forth people to join ASCII if they want, but don't leave screens and blocks!

Sincerely,

Pierluigi De Rosa

Via Nicola Parisio 4/C
87100 Cosenza, Italy

Name That Architecture...

Dear Editor,

Articles now appearing on Forth-related processors have many ways of naming these items. For example, I have seen the terms RISC, Forth Engine, and Stack Machine. While these are descriptive, a

FORTH SOURCE™

WISC CPU/16

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

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

MVP-FORTH

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

MVP Books - A Series

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

MVP-FORTH Software - A trans- portable FORTH

- MVP-FORTH Programmer's Kit including disk, documentation. Volumes 1, 2 & 7 of MVP Series, FORTH Applications, and Starting FORTH. IBM, Apple, Amiga, CP/M, MS-DOS, PDP-11 and others. Specify. \$195
- MVP-FORTH Enhancement Package for IBM Programmer's Kit. Includes full screen editor & MS-DOS file interface. \$110
- MVP-FORTH Floating Point and Math IBM, Apple, or CP/M, 8". \$75
- MVP-LIBFORTH for IBM. Four disks of enhancements. \$25
- MVP-FORTH Screen editor for IBM. \$15
- MVP-FORTH Graphics Extension for IBM or Apple \$80
- MVP-FORTH PADS (Professional Application Development System)
 An integrated system for customizing your FORTH programs and applications. PADS is a true professional development system. Specify Computer: IBM Apple \$500
- MVP-FORTH Floating Point Math \$100
- MVP-FORTH Graphics Extension \$80
- MVP-FORTH EXPERT-2 System for learning and developing knowledge based programs. Specify Apple, IBM, or CP/M 8". \$100

Order Numbers:
800-321-4103

(In California) 415-961-4103

FREE
 CATALOG

**MOUNTAIN VIEW
 PRESS**

PO DRAWER X
 Mountain View, CA 94040

better naming approach will have a few benefits. I propose that processors that run a Forth kernel, whether hard-wired or by programming in ROM or in microcode, have a standard, family name. There are many ways of doing this.

Following the style used by the mainstream, they can be labelled Forth Instruction Set Computers (FISC). This form parallels that used by other popular processor architectures: Complex Instruction Set Computers (CISC) and Reduced Instruction Set Computers (RISC).

A more technical term could be used: Threaded, Interpretive Stack Machines (TISM). This term is more precise and broadly applicable, and can even be used to describe processors related to Forth's architecture; for example, a Writable Instruction Set Processor. [Or WISC Technologies' Writable Instruction Set Computers. —ed.]

Another alternative is to honor Charles H. Moore, the creator of Forth and co-designer of a commercial Forth engine (the Novix NC4000), by naming the "two-

stack, two-pointer, 4-space machine" after him. Unfortunately, the most direct term, Moore Machine, is already in use in connection with state-machine theory. Maybe someone can come up with something else.

The term "Forth engine," while it is still applicable to a FISC, does not seem correct, since Forth itself is undefined. Further, Forth's extensibility has not been translated to hardware extensibility. [You has better look at those WISC machines, Jose.—ed.] Perhaps, when someone puts a Xilinx logic cell array; a writable, threaded, interpretive stack machine; 8K EEPROM; and an LCD with nano-keyboard on one tiny chip, and this anything chip leisurely chugs along at 33 MIPS, we will have an interactive, real-time engine.

Sincerely,
 Jose Betancourt
 85 Arlo Road #1A
 Staten Island, New York 10301

```

: (LOCAL)
  R> R> ! ;           \ restore variable address and value
from return stack

: LOCAL ( adr - )
  R> SWAP              \ save top return address
  DUP @ SWAP >R >R    \ put variable address and value on
return stack
  ['] (LOCAL) >BODY ADDR>S&O >R \ exit via (LOCAL)
  >R ;                \ restore top return address to con-
continue current definition
  
```

Figure One. Hansen's local variables for PC/FORTH+.

```

VARIABLE VAR
: N! ( n — n! )
  VAR LOCAL
  DUP VAR !
  1- DUP 0> IF RECURSE ELSE DROP 1 THEN
  VAR @ * ;
\ 10000 times 12 N! in 45 sec, with PC/FORTH+.
  
```

The simpler word n! is much faster, without the local variable:

```

: n! ( n — n! )
  1 SWAP 1+ 1 ?DO I * LOOP ;
\ 10000 times 12 n! in 10 sec.
  
```

Figure Two. Sample use of LOCAL.

A FREE SPIRIT: WHERE TO FROM HERE?

GLEN B. HAYDON - LA HONDA, CALIFORNIA

A free spirit is, perhaps, the single most important trait of Forth users. Each user seems to have his own philosophy, religion, and brand of the language. Each certainly has his own expectations. This free spirit made Forth what it is and, at the same time, led to its lack of general acceptance. Programmers using other languages have never experienced such a free environment.

Many Forth programmers need to eat but find little acceptance of their ideas. Some have been able to use variations of Forth in their work place, but not many. Many accept Forth as their avocation, and program for a living in other languages.

The efforts of some vendors and application programmers to develop a common basis is in progress. But already some vendors have clashed. One wants exactly the opposite of what another wants. By codifying a language, a free spirit is stifled.

For some years, I have attempted to understand the free spirit of the creator of Forth. Chuck Moore's concepts provide the fundamental basis of this language, which he named Forth. He wants control over the hardware. He wants to keep the program small, because a small, simple program is efficient. Over the years, he has heard many suggestions; most of them he has discarded.

Han Nieuwenhuyzen objected to giving the programmer access to all the hardware. For years he has used file structures of his own devising in his Forth implementations. For years, few implementors listened to his suggestions.

The Forth-79 Standard excludes from

the required word set any primitives that access the hardware. The Forth-83 Standard continues this movement away from the basic hardware. Many users want to run other programs on their hardware; they use programs daily which are not a part of Forth. But at the same time, they have applications they wish to program in Forth. All sorts of compromises are made.

The free spirit of Forth implementors has prevented programmers, conditioned to the constraints of conventional languages and operating systems, from adhering to a Forth standard. Is there a common thread running through all variations of Forth?

*There is no reason for
Forth programmers to be at
odds.*

Various new languages (e.g., Fifth, Reptil, Stoic, Urth) have one thing in common with Forth: they are threaded, interpretive languages. As a matter of fact, even Microsoft has come to recognize the advantages of a threaded, interpretive language. Their latest version of Quick Basic is implemented in such a manner. And they indicate they will use a threaded interpreted implementation of C and other packages in the future.

There is no reason to throw out the baby with the wash. A subset of Forth is coming of age, as threaded, interpretive languages are more widely adopted. Loeliger's book, *Threaded Interpretive Languages* was before its time. Let threaded, interpretive

languages grow.

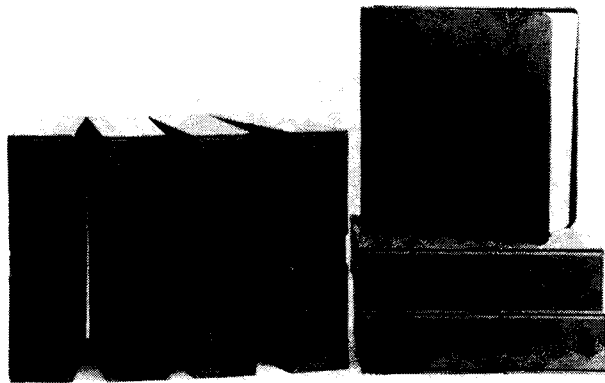
There is no reason for Forth programmers to be at odds with one another. We could focus on a subset upon which we agree, and apply our free spirit to other areas.

Perhaps the Forth Interest Group should evolve. It could expand its horizons to include all variations of threaded interpreted languages; perhaps it could even include a threaded, interpretive BASIC. This would be a move away from the Forth Chuck gave us along with his concern for the hardware. (Since, after all, Chuck is the creator of Forth, maybe such a new direction should be given a new name.)

For many years, I have considered Forth as Chuck's child. I still do. But I am not constrained to do everything in Forth. I have an excellent word processor that is not written in Forth. (At least, I don't think it is.) I use several desktop publishing programs, which I know are not written in Forth. I use my computers for other things, too.

Several individuals at the recent FORML conference, each with his own concepts, presented "free-spirit" language implementations. Tom Zimmer, in the FIG spirit of public-domain shareware, provided attendees of FORML a Forth without BLOCKS. His implementation can serve as a command-processor shell for IBM-compatible processors, with access to DOS functions. He has included an editor that is very much like WordStar, but any word processor producing ASCII files can be used. I am sure Tom's release is a subset of what he now uses commercially.

TOTAL CONTROL with LMI FORTH™



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

For Development: Interactive Forth-83 Interpreter/Compilers

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

For Applications: Forth-83 Metacompiler

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

For Speed: CForth Application Compiler

- Translates "high-level" Forth into in-line, optimized machine code
- Can generate ROMable code

Support Services for registered users:

- Technical Assistance Hotline
- Periodic newsletters and low-cost updates
- Bulletin Board System

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

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

Overseas Distributors.

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

It points out a different, possible direction for free spirits.

Mitch Bradley has long urged the community to give up blocks. He has provided many file-oriented routines. He is constrained to use his processors for other programs than Forth. The people he works with have trouble with some of the primitive ways of Forth.

Forth adherents adopt their own concept of Forth with a religious fervor. It is small wonder that Forth, whatever it is, is not accepted by the rest of the world. But Microsoft has adopted a threaded, interpretive language; so has Adobe with their PostScript, another of Forth's offspring. This fundamental aspect of Forth is being adopted widely.

With the evolution and expansion of the Board of Directors of the Forth Interest Group, which took place at the recent national convention, it may be time to reassess the organization's focus. By recognizing the common denominator in Forth, and the direction taken by larger software houses (threaded interpreted languages), we have a common direction that is less loaded with emotion.

It might even be appropriate to emphasize the commonality in the community by modifying the name of its regular publication. Maybe a subtitle would serve to indicate a change of emphasis. I am not proposing that the organization and its publication change their names. But, just maybe, that extreme step could be considered. Any such decision is in the hands of the Board of Directors.

In any case, I feel the community could be drawn together by focusing on what the members have in common — a threaded interpreted language. Let Forth users rally around their common convictions. There is plenty of room for difference without emotional conflict. Improvements will come through the efforts free spirits: Chuck Moore, Hans Nieuwenhuyzen, Tom Zimmer, Mitch Bradley, and many others have contributed because of their free spirits, not because of any imposed standard.

Glen B. Haydon is widely known as the implementor of MVP-FORTH and, along with Phil Koopman, Jr., creator of the WISC CPU/16 and CPU/32 processors.

MODULE MANAGEMENT

ALAN T. FURMAN - SANTA CLARA, CALIFORNIA

I originally became hooked on extensibility years ago while programming in SAIL, which heavily supports macros. In SAIL, one can have files of macro definitions, both private stock and community contributions, and make them available for use in a program with the compiler-directive statement

```
require <filename>
```

where the file could itself contain more requires. A well-designed set of extension packages permits tiny, highly expressive programs, with the "require" mechanism looking after the bringing-in of macros and their hierarchical dependencies. The "include" statements of Pascal and C work similarly. I decided that Forth needed such a facility. I also wanted to be able to refer to source code packages by name, rather than by numerical address.

The module management system described here provides, essentially, all the convenience of the SAIL "require" in seven lines of source code. It is more limited, but simpler, than a prior scheme¹ described by Michaloski. It runs under, and is easily transported to, any Forth system running under the screen model, whether true physical blocks or logical 1 kbyte records in a DOS file. It does not in any way depend on a DOS file system as such.

What It Does

In brief, the module management system is a way of giving a symbolic name to a group of screens that contain generally reusable source code. This name is a Forth

word whose action is to guarantee presence of the package in the dictionary, to be used by applications. (The system involves redefinition of the module name word. Therefore, it works only when the module name word is interpreted, by the text interpreter, from the keyboard or from a screen being loaded. It will not work correctly when called from inside a colon definition.) An example follows.

Suppose that a module — a named set of words — called TRIGONOMETRY exists on disk, and one or more of these words is needed by a new word about to be defined. Just type

```
TRIGONOMETRY
```

and commence defining the new word. If the trigonometric functions are not in the dictionary, TRIGONOMETRY causes them to be compiled from the disk. If they are already in the dictionary, TRIGONOMETRY is a no-op (nothing happens). That is, the effect of TRIGONOMETRY is to ensure that the trigonometry package is in the dictionary, available to be interpreted or compiled into a higher-level word.

The Forth user is freed from two chores: remembering the numerical addresses of the trigonometry package source, and keeping track of whether they have been compiled into the dictionary yet. If a module depends on another, just embed the lower-level module's name in the higher-level module's source. For example, a GRAPHICS module may invoke TRIGONOMETRY because it uses the latter's words. A ROBOTICS module could use TRIGONOMETRY as well. Now suppose that a

robot simulator requires both GRAPHICS and ROBOTICS. During the compilation of GRAPHICS, the trigonometry words will be compiled by TRIGONOMETRY. When ROBOTICS compiles, the action of TRIGONOMETRY will be a no-op, thereby preventing redundant compilation. It is this "smart" behavior that makes nesting of modules a true convenience.

The module name word also acts as a place-marker for reclaiming dictionary space; typing

```
FORGET TRIGONOMETRY
```

shrinks the dictionary back to just before the trigonometry package.

How It Works

The module management system has three parts. First, the module management wordset

```
: MODULE ( block#)
  CREATE , DOES> @ LOAD ;

: LOADMAP: ( -- address)
  CREATE HERE -1 ,
  DOES> @ ABORT" MODULE
  ERROR" ;

: LOADED ( address)
  0 SWAP ! ;
```

which is compiled from the disk immediately upon booting Forth.

Second, the module declarations, for example:

Australia

Announcing a group travel plan to attend the first Australian Forth Symposium in Sydney May 19th & 20th, 1988 and World Expo 88 in Brisbane May 21-23, 1988. Other group events include guided sightseeing tours and a visit to Lamington National Park located in South East Queensland. Optional travel additions can be arranged.

Australian Forth Symposium

Charles Moore, Forth's inventor, is the keynote speaker at this event. The symposium has been initiated by a group of professionally based Forth users from both industrial and academic organizations who believe that the language should be more widely known and used by the professional community. The focus is Forth as a programming system for productivity. The first day will feature presented papers and demonstrations to show what can be done in Forth, and how quickly working applications can be developed. The second day will offer a choice of special interest Workshops, with instruction and hands-on experience. An exhibition will be open both days.

World Expo 88

Australia will host one of the world's biggest celebrations in 1988. **World Expo 88**, the international highlight of Australia's Bicentenary, will be held in the heart of Queensland's capital, Brisbane, from April 30th to October 30th. **World Expo 88** will be the largest single event in the nation's history with an estimated attendance of almost eight million from throughout Australia and other countries. More than 30 nations and 20 corporations will showcase their achievements under the theme "Leisure in the Age of Technology". The 100 acre Expo site is ideally located on the South Bank of the Brisbane River, only 100 meters from the heart of the Sunshine City of Brisbane.

Group Travel Itinerary

Depart San Francisco Monday May 16, 1988 arriving Sydney May 18th. Attend the **Australian Forth Symposium** May 19th and 20th with local tours arranged for non-conference guests. Fly to Brisbane on May 21st and visit **World Expo 88** through May 23rd; local tours will also be available. Travel to **Lamington National Park** May 24th with accommodations in a mountain lodge through May 26th. Return to Brisbane on May 27th and take the return flight to San Francisco.

Reservations and Information Brochure

Contact the **Forth Interest Group**, P. O. Box 8231, San Jose, CA 95155, telephone (408) 277-0668.

```
100 MODULE TRIGONOMETRY
120 MODULE GRAPHICS
140 MODULE ROBOTICS
```

etc.

HERE FENCE !

which are compiled right after the module management wordset. They create a group of words in the dictionary that serve as a kind of "directory" of modules.

Third, the modules themselves. Each module begins with a "load map" screen, whose number is the number incorporated in the definition of the module word. This loadmap invokes the words LOADMAP : and LOADED and also causes the remaining screens (which contain the module's actual source code) to be loaded. The general outline of the loadmap is:

```
LOADMAP : <modulename>
<loading of source screens>
LOADED
```

This is how the loadmap for TRIGONOMETRY (screen 100) might look:

```
LOADMAP : TRIGONOMETRY
101 LOAD 102 LOAD 103 LOAD
LOADED
```

The top line is not a comment, but it eliminates the need for one.

Now consider the case where the system has been booted and the module management wordset and the module declarations have been compiled. The word TRIGONOMETRY is defined such that its action is 100 LOAD. Therefore, when the word TRIGONOMETRY is interpreted by the text interpreter (typed at the terminal or read off a screen), its action will be to load screen 100. This screen redefines TRIGONOMETRY to be a no-op (causing an inconsequential "Redefined" or "Isn't Unique" message), and compiles the code on screens 101 through 103. The next time TRIGONOMETRY is interpreted, it will act according to its new definition, which is a no-op.

As mentioned, modules can be nested. For example, screen 120 might look like this:

```
LOADMAP : GRAPHICS
TRIGONOMETRY
121 LOAD 122 LOAD 123 LOAD 124
LOAD
LOADED
```

Screens 121-124 can thus assume that the trigonometry wordset will be available.

Security

Two provisions in the module management system that deal with security remain to be discussed.

First, LOADMAP : does not exactly redefine the module name to be a no-op. Indeed, it defines a word that will abort upon execution. It leaves on the stack a pointer to the location of the -1 flag so that LOADED can subsequently overwrite it with a 0. Only then has the module name been redefined as a no-op. The reason for this two-stage process is that a module should be either completely missing from the dictionary, or completely compiled. If compilation is interrupted (as by a compile error), the module is unusable. The combination of LOADMAP : and LOADED prevent the user from attempting to use a fragmented module. It also prevents catastrophic infinite-loop mutual recursion in case two modules try to "require" each other.

The second unexplained provision is the phrase

HERE FENCE !

which makes whatever precedes it unFORGETtable. The very first word added to the dictionary when a module is compiled is the redefinition of <modulename> performed by LOADMAP :. As a result, one can cleanly wipe out a module from the dictionary with

```
FORGET <modulename>
```

which works fine if the module is there. If the module is not, then the most recently compiled word named <modulename> is somewhere in the "directory" of modules, which FORGET would partially destroy, if it could reach the word. Having the "directory" protected in its entirety saves the user from having to keep track of a module's presence in the dictionary. Trying to FOR-



NGS FORTH

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

STANDARD FEATURES INCLUDE:

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

A COMPLETE FORTH
DEVELOPMENT SYSTEM.

PRICES START AT \$70

NEW ◀ HP-150 & HP-110
VERSIONS AVAILABLE



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

**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 **four to 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 Forths 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**, **CRASH-FORTH** and, for TRS-80, **FREEWAY**), board games (**OTHELLO** and **TIC-TAC-FORTH**), and a top-notch **CRYPTO-QUOTE 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. Better yet, call 617/653-6136.**

MMSFORTH

and a free gift!

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
DATAHANDLER 59.95
DATAHANDLER-PLUS* 99.95
EXPERT-2 69.95
UTILITIES 49.95

*Single-computer, single-user prices; corporate site licenses from \$1,000 additional. 3 1/2" format, add \$5/disk; Tandy 1000, add \$20. Add S/H, plus 5% tax on Mass. orders. DH+ not avail. 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. \$21.95*
THINKING FORTH 16.95
Many others in stock.

MILLER MICROCOMPUTER SERVICES
61 Lake Shore Road, Natick, MA 01760
(617/653-6136, 9 am - 9 pm)

presence in the dictionary. Trying to **FORGET** an uncompiled module will then result in a harmless error message.

Further Enhancements

The module management system is shown in its final form in Listing One. The first screen may be used verbatim (the other screens are usage examples). Normally, it is loaded first thing upon booting the system; it will often be the only screen number the user has to memorize. In the process, the module directory on the following screen will be compiled into the dictionary and protected by **FENCE**, and a constant **MODULES** will be the screen number of the directory source. Type

MODULES LIST

to list the modules.

The word **THRU** (see Appendix) makes loadmaps more concise. Given the first and last screen numbers, **THRU** loads the inclusive range of screens. The word **+B** converts a relative screen number to an absolute one. Using it in loadmaps makes modules relocatable on the disk. The loadmap in the **TRIGONOMETRY** example given above can be reduced to the form shown as Screen 100 in Listing Two. The module can now be moved (as a unit, including the load map) to another region of the disk. No editing is required, either in the module itself or in modules that require it (the name **TRIGONOMETRY** remains as before). However, the original declaration

```
100 MODULE TRIGONOMETRY
```

on line 2 of Screen 81 will have to be edited (to reflect the new address of the load map) and recompiled.

Whenever the module declaration screen is edited (due to the addition, deletion, or moving of a module), the module directory in the dictionary must be compiled anew from the disk. It is first necessary to **FORGET** the prior directory, but it is protected by **FENCE**. The following procedure will recompile the directory:

```
MODULEMARK FENCE !  
FORGET MODULEMARK  
80 LOAD
```

and in the process, delete everything else compiled in since boot-up. It may be just as practical to reboot the system and type

```
80 LOAD
```

as usual. Both alternatives seem troublesome, but seldom need to be performed. Once a module has reached final size, acquired a reliability record, and received heavy reuse in applications, it is unlikely to move about.

Conclusion

A system has been described for naming reusable, Forth source code packages, reminiscent of the "require" and "include" facilities of other languages, in which the package name becomes a word that compiles the package into the dictionary, as needed. It is easy to use and install, and supports hierarchies of packages.

The extreme simplicity and degree of control afforded by screens makes this system very easy to port, whether in a true native Forth or an emulated one with a single "screen file."

References

1. J. Michaloski, "A Forth Profile Management System," *Journal of Forth Application and Research*, Vol. 2, No. 3, p 63-75 (1984).

Appendix: System Dependencies

FENCE (-- address)

The commonest name of a variable that points to the last protected word in the dictionary. Unfortunately, the choice of which of a word's fields **FENCE** points to is unstandardized. The value of the constant **MODULEMARK** as generated in Screen 80 should reliably defeat the protection of **MODULEMARK** from **FORGET** when written into **FENCE**.

The following words are easily defined if a system lacks them. Their definitions are best put in screen 80, between the definitions of **MODULES** and **MODULE**.

```
: THRU ( firstscreen lastscreen -- )  
  1+ SWAP DO LOAD LOOP ;
```

```
: +B ( relativeblock -- absoluteblock )  
  BLK @ + ;
```

In a system without ABORT" replace

ABORT" MODULE ERROR"

with

IF ." MODULE ERROR" ABORT
THEN

Listing One. Source code for module system; assumes that module declarations are on screen 81.

```
SCREEN # 80
0 ( MODULE SYSTEM -- LOAD ME FIRST AFTER BOOT-UP )
1 HERE 1- CONSTANT MODULEMARK ( USE FOR ERASING MODULE
WORDS )
2 1 +B CONSTANT MODULES ( DECLARE MODULES IN FOLLOWING
SCREEN )
3
4 : MODULE ( BLOCK#)
5   CREATE , DOES> @ LOAD ;
6 : LOADMAP: ( - ADDRESS)
7   CREATE HERE -1 , DOES> @ ABORT" MODULE ERROR" ;
8 : LOADED ( ADDRESS)
9   0 SWAP ! ;
10
11 MODULES LOAD   HERE FENCE !
```

Listing Two. Examples of module usage.

```
SCREEN # 81
0 ( MODULE DECLARATIONS FOR EXAMPLES )
1 85 MODULE SCREENEDITOR
2 100 MODULE TRIGONOMETRY
3 120 MODULE GRAPHICS
4 140 MODULE ROBOTICS
5   .
6   .
7   .

SCREEN # 100
0 LOADMAP: TRIGONOMETRY ( EXAMPLE)
1 1 +B 3 +B THRU ( ACTUAL CODE ON SCREENS 101-103)
2 LOADED

SCREEN # 120
0 LOADMAP: GRAPHICS ( EXAMPLE-NESTED MODULES)
1 TRIGONOMETRY
2 1 +B 4 +B THRU ( ACTUAL CODE ON SCREENS 121-124)
3 LOADED
```

BRYTE FORTH

for the

INTEL 8031 MICRO- CONTROLLER



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

1987 FORTH NATIONAL CONVENTION

JERRY SHIFRIN - MCLEAN, VIRGINIA

The Forth Interest Group (FIG) held its ninth annual convention on November 13-14, the day after the second ANS Forth standards meeting, in San Jose, California. This is the first FIG convention I've attended, and I found it to be another enjoyable Forth experience.

The theme was the "Evolution of Forth." Ironically, a number of people seemed concerned with whether or not Forth was dying. In the Forth philosophy session, Alan Furman pointed out that it's difficult to convince people you have a better mousetrap (Forth) so they'll beat a path to your door, when the neighboring house (C) has a freeway running through it! Several other speakers seemed to echo this concern. In a response of sorts, Chuck Moore pointed out that it really doesn't matter if a lot of people are using Forth, or even if anyone else is, since he can improve his own productivity with it.

This is not to imply that the meeting was downbeat — there were numerous discussions on new and exciting developments in the Forth world: new Forth chips, new implementations, new books, and many new applications.

Friday

Several sessions were dedicated to the history of Forth, with presentations from many of its pioneers: Chuck Moore, Elizabeth Rather, Bill Ragsdale, Kim Harris, and so on. In this paper, I'll only comment on the technical sessions.

Charles Johnson gave an interesting talk on their MISC (Minimal Instruction Set Computer) chip, which is still in devel-

opment. It seems to be an inexpensive implementation of a Novix-like architecture. So far, the MISC chip has only been run in simulation.

I found the philosophy panel discussion to be the highlight of the conference. This included Wil Baden (who showed up in his philosopher's robes), Glen Haydon, Chuck Moore, Alan Furman, and Howard Pearlmuter. One of the main themes of this discussion was the "less is more" idea, that we should return to minimal Forth systems. Wil Baden described some of the current implementations as "garbage pail Forths," as in garbage pail pizza (with everything). Glen Haydon pointed out that Forth programmers need to decide whether they want to be free spirits or professionals, whether Forth is their vocation or avocation, and to act accordingly. Howard Pearlmuter "performed" (the only remotely appropriate verb) a metaphor on Forth. Variousy entitled "Four Thoughts," "Forethoughts," and "Forth Oughts," he described Forth as a tree with its roots in dirt and sand (silicon) and it leaves and branches reaching out to the sunshine (people). He noted that Forth allows us to get close to the silicon, but that we should pay more attention to the users.

Friday wrapped up with a "Point/Counterpoint" discussion (renamed to "Count, Pointer, Count") with Mitch Bradley, Mike Perry, and Martin Tracy.

Saturday

Saturday continued the oral history of Forth with presentations from the fig-FORTH and F83 implementation teams.

Guy Kelly gave a history of the Forth-83 Standard.

George Nichols gave a presentation on their NOVIX-based PC-RISC system, running multiple Novix co-processors in an IBM PC.

Along with Martin Tracy, Bob Smith, and Larry Forsley, I participated in a panel discussion on the ANS Forth effort. This wasn't a terribly crowded session, but the response seemed positive overall. People were happy that the effort was underway; they approved of using the Forth-83 Standard as the basis; and were mainly concerned that the standard include some optional extensions, floating point being mentioned most often. When introduced, the ANS Forth members received what can only be described as a smattering of applause.

Gary Feierbach described their Super-8 system, optimized for Forth and including Forth in ROM. It directly executes NEXT, DOCOL, and EXIT in a single instruction (though each instruction requires multiple machine cycles). The chip is expected to sell for \$7 in single unit quantity. It supports a 64K program space and a 64K data space. The Zilog development board costs \$88 and the Inner Access ROM costs \$65. The processor has an effective cycle time of 10 MHz.

There was a well-attended panel discussion on the GENie Forth service. Dennis Ruffer, Scott Squires, Marlin Ouverson, and Lori Chavez described various aspects of the available facilities. Alan Furman described the user interface as "technologically nostalgic."

The annual meeting noted the resignations of Thea Martin and Kim Harris from the FIG Board of Directors, the continuance of Robert Reiling and Martin Tracy, the re-election of John Hall, and the election of Wil Baden, Bob Smith, Dennis Ruffer, and Terri Sutton.

Other sessions included a FIG Chapters breakfast, a second talk on the MISC chip, a talk from Phil Burk on HMSL (an object-oriented, Forth-based music language), Glen Haydon on the 32-bit WISC engine, Lori Chavez on the Unstable Flying Wing Project, a roundtable on 32-bit applications, and one on Forth in education.

Another highlight of the conference was Chuck Moore's "Fireside Chat."

The conference was capped off with an after-dinner speech by Dr. Robert Trelease on "Brains, AI, and Forth." He gave an overview of current Forth activity in AI and expert systems (see his article in the October 1987 issue of *AI Expert*), and went on to discuss the current state of affairs in understanding how the brain operates, and neural network technology.

Notes

As always, there is at least as much going on outside the sessions as there is on the inside. Here are some of the notes I made:

There is a new Forth BBS for our friends to the North: a Vancouver, British Columbia board at 604-434-5886. Zafar Essak is the operator.

Martin Tracy has begun developing an iconic (picture-oriented) programming language. Where does he find the time? Martin's next *Dr. Dobbs* column is scheduled for December 1987, and should include some of the material from the ECFB discussion of strings.

Gary Betts mentioned that his company (Saba Technologies) is planning to upgrade their inexpensive document scanner (programmed in Forth) to take advantage of the Novix chip.

I saw an actual demonstration of Harvard Softworks' and Softmills' GigaForth. It looked like a powerful, well-thought-out implementation. First customer shipment is scheduled for January 1988. It sells for \$245 as an add-on to HS/Forth (\$395). Another add-on, Gigaloom and Rosetta, is still in development; this is intended to

allow programmers to develop and link modules from a variety of languages within a single environment.

HyperForth! The current rage in the personal computer arena is something called hypertext, a way of scattering your data all over, but still being able to retrieve it quickly. Well, the inventor of hypertext, Ted Nelson, has been involved with the Forth community for many years. Bob LaQuey has concluded that they are kindred systems, and has begun designing an integration of the two concepts into a single system. (He will presented a paper on this at FORML.)

A new book of interest from MIT Press: *Cellular Automata Machines* by Tommaso Toffoli and Norman Margolus (\$30, 260 pages). Presents the theory of cellular automata and develops a language for describing cellular automata rules, CAM Forth, which has been implemented to support a CAM board in an IBM PC.

Dr. C. H. Ting's Offete Enterprises has a few new additions to its catalog of Forth material: *Forth Notebook*, Volume II (\$25: ROMable F83, 8086 and 68000 disassemblers, parallel processing, array processing, neural network simulation, etc.); *F83 Reference Manual* (\$10); the *More on NC4000* series is now up to 5 volumes (total cost for the series is \$70).

Guy Kelly is selling his portable Forth editor for \$20, which includes his PD PC-Forth implementation and support for LMI, F83, MVP, UniForth, and his own system.

An Australian Forth Symposium is scheduled for May 19-20, 1988 at the NSW Institute of Technology. Chuck Moore will be the keynote speaker. Contact Jose Alfonso, NSWIT, P. O. Box 123, Broadway NSW 2007.

As some of you might have noticed recently, some of the mail sent to the Institute for Applied Forth Research (publisher of *The Journal of Forth Application and Research*, or JFAR) was returned without explanation or with something along the lines of "moved, left no forwarding address." Well, it turns out that they did move, but the post office decided not to forward their mail. Larry Forsley reports this wasn't discovered for several weeks — not until someone handed him an envelope with the post office's practical joke on it. According to Larry, this problem has been resolved,

but here are the correct addresses:

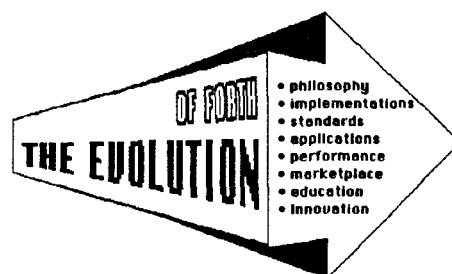
For subscriptions: Total Information, 844 Dewey Avenue, Rochester, NY 14613. (Total Information is one of those subscription fulfillment services.) The new address for The Institute for Applied Forth Research is: 70 Elmwood Avenue, Rochester, NY 14611.

Manuscript submissions should be mailed to: Jim Basile, 17 Target Rock Drive, Huntington, NY 11743. (Jim is the new Editor-in-Chief of JFAR. Larry Forsley is the publisher, Mahlon Kelly is the US Editor, and Hans Nieuwenhuyzen is the European Editor.

I don't know if it's the hacker mentality, but there seemed to be hostility directed towards some of the Forth vendors, mainly FORTH, Inc. There appeared to be some gloating that the public-domain, Laxen & Perry F83 implementation had forced Forth vendors to provide more complete systems. Even if true, I found the attitude less than attractive.

Computer Literacy Bookstore and Fry's Electronics can almost justify the trip to San Jose on their own. Computer Literacy has, by far, the best selection of computer books I've ever come across (in D.C., try Reiter's for one almost as good). Fry's is a supermarket with food, audio/video, computers, software, and electronics parts. Take a shopping cart and your credit card.

Jerry Shifrin is a prolific talent; see more of his work in this issue's "ANS Forth Meeting Notes."



The Evolution of Forth: Forth is entering its second decade; where will it be after ten more years? At this year's Forth National Convention, learn the expert opinions of market leaders, implementors of Forth on the new generation of machines, respected Forth theorists, and programmers of significant Forth applications. Should you prepare, or just despair? Find out how Forth's future could affect you!

ANS FORTH MEETING NOTES

JERRY SHIFRIN - MCLEAN, VIRGINIA

TC Meeting 1

The second meeting of the ANS Forth Technical Committee (TC) was held on November 11-12, 1987 in San Jose, California. Local arrangements were provided by Bill Ragsdale and the Forth Interest Group. The TC still does not have an official Chair and Vice-chair. Some candidates for these positions were unable to get approval from their companies for sufficient time away from work. As a result, the meeting was officiated by Elizabeth Rather as Acting Chair, with Martin Tracy as Acting Secretary. The Chair and Vice-Chair positions are still open if anyone wishes to volunteer. ANSI/CBEMA decided not to appoint permanent officers until they had more people to choose from. At the moment, candidates for Chair are Charlie Keane of PPI and John Dorband of GSFC. Ray Duncan is the only candidate for Vice-chair.

Most of the attendees from the August meeting were present here with the exception of Charlie Keane, David Petty, and the ANSI/CBEMA folks. New to this meeting were Wil Baden, Andy Kobziar of NCR (user), John Gotwals of Purdue (user), and John Stevenson (user). Attending as an observer was Dennis Ruffer (GENIE SYSOP) from Allen Test Products.

We received a letter from our CBEMA liaison, John Kurihara, congratulating us on the quality of our efforts to date and complimenting Ray Duncan on the minutes of the first meeting.

The TC approved the meeting schedule without objection. The next meeting is scheduled for February 10-12, 1988 in

Southern California, to be hosted by Elizabeth Rather. Subsequent meetings are planned for Rochester, New York (Larry Forsley); Beaverton, Oregon (Gary Betts); and Washington, DC (Jim Rash). Chuck Moore suggested that meeting hosts attempt to find sites in surroundings more pleasant than hotels. Ms. Rather agreed to look into holding the February meeting on Catalina Island. Larry Forsley said he was investigating the availability of a mountain retreat. Meeting plans will be announced when available. Anyone interested in attending should contact the appropriate meeting host for information.

*It's to your advantage that
your products can be
stamped "ANSI Forth"*

A form for technical proposals was discussed. The documentation committee was directed to add a cover sheet with detailed instructions, and to remove any fields intended for internal TC usage. This was to have been circulated and voted on by mail ballot within two weeks of the meeting.

Next was a report from the Research Committee on current Forth practices. A total of 274 surveys were mailed out, but only 24 responses were received. Of these, only 14 indicated 200 or more users (required for consideration of "common usage," by a previous vote). The results from this survey were rather interesting: respondents who differed from the Forth-83 standard did so in the areas of word addressing,

32-bits, lack of floored division, and lack of disk commands. Most respondents offered extensions in the areas of strings, multi-tasking, graphics, floating point, etc. Suggestions from respondents varied widely, but a number of people thought the standard should be layered (allow optional, standard extensions), and that it needed to deal with 32 bits, floating point, OS interface, strings, graphics, and ROMability. The Research Committee was directed to make another attempt to obtain responses from "major" vendors who had not returned their questionnaires.

The Technical Subcommittee (TSC) next reported on areas of consensus and controversy with respect to the Forth-83 Standard. Surprisingly, there were a few areas where the TSC was in unanimity on keeping certain items from the 83 standard: DUP, DROP, OVER, SWAP, >R, R>, AND, OR, XOR, +, -, ABS (ABS was later found to be controversial), 0=, 0<, =, U<, @, !, and the ASCII collating sequence. Everything else in the 83 standard had either major or minor controversy associated with it.

Elizabeth Rather read letters from Larry Forsley and Guy Kelly on the IEEE Forth Standard activity. We haven't received official word yet, but it appears that the IEEE Forth proposal has been withdrawn in acknowledgment of the openness of the ANSI/CBEMA effort. One major benefit from this controversy is that members of the IEEE Computer Society may attend ANS Forth TC meetings without paying any fees. Since Computer Society membership costs about \$20, there is a clear advantage to this alternative approach.

The TC then had a fairly lengthy discussion on the Technical Proposal process. Briefly, proposals are to be sent to the secretary (Martin Tracy at FORTH Inc.). The secretary will pre-filter proposals, returning any in obvious need of work. He will distribute remaining proposals to the TSC, which will return them to the TC with a recommendation to adopt, reject, return for more information, or table. If the TC agrees to adopt, it would be sent to the Documentation subcommittee — who will develop the final language — then to the TC, who will either ratify or return it. The TC voted (14-0) to allow a proposal to be tabled indefinitely. The secretary was directed to maintain a database (in a format of his choice) of all proposals.

I pointed out a problem, in that Technical Proposals were supposed to be submitted in the form of updates to the Basis document (this is the Forth-83 standard initially, but it evolves into the draft standard as updates are made). However, we are not allowed (by CBEMA) to make the Basis document publicly available. The sense of the TC was that, initially, people should submit proposals as updates to the Forth-83 Standard. Later, they will have to develop their proposals in conjunction with a TC member.

The TC voted (12-3) to maintain a public status list of all proposals. I will be maintaining this on the MCI Mail ANS Forth Bulletin Board. The TC also voted (13-2) to similarly publish all proposal abstracts received on electronic media.

The TC then adjourned to allow for subcommittee meetings.

TSC Meeting

The TSC (Technical Ad Hoc Subcommittee) immediately convened to begin deliberations. Greg Bailey was still Acting Chair and Martin Tracy continued as Acting Secretary.

The first order of business was to discuss the process for dealing with Technical Proposals within the TSC. The final conclusion, to the best of my understanding, is that proposals found to be controversial will be directed to a "magnet" assigned to each major area of controversy. The magnet will collect comments from the entire TC and circulate them for

review. The goal here is to allow work to continue expeditiously outside formal TC meetings. Non-controversial proposals go directly to the TC with a TSC recommendation.

There were discussions on word or cell size (non-controversial) and signed division (controversial).

Next were discussions on voting rights. I have two conflicting notes here: one says that only formal members of the TC (including alternates and observers) may vote on TSC issues; the other states that, since the TSC is a totally ad hoc committee, anyone who attends can vote. I'll clarify this ruling when/if I can.

There was a discussion on getting a TSC secretary. Martin Tracy cannot continue, since he is too busy as the TC secretary. No one else cared to volunteer. Don Colburn said he'd do it as a last resort, but that he'd make us pay for it! (Don, who was the secretary of the Forth Standards Team, felt he had suffered enough.) Finally, I agreed to take on the work, but I cautioned the group that I would not be able to attend all of the meetings.

We agreed that MCI Mail was not a suitable facility for holding TSC discussions between meetings (MCI Mail Bulletin Boards are mainly oriented towards posting announcements). We then discussed the advantages of GENie vs. CompuServe. Don Colburn offered to make available a section of his CompuServe Forth conference, and Dennis Ruffer offered to do the same on FIG's GENie Forth conference.

There was discussion and vote on Elizabeth Rather's cell-size proposal (eliminating references to 16-bit words). This was found to be non-controversial by a vote of 17-2-1-0 (strongly in favor, prefer inclusion, prefer exclusion, strongly in favor of exclusion) and was referred to the TC. From the vote, I conclude that at this point anyone present was eligible to vote.

As secretary, Martin Tracy had received four proposals to date. Naturally, none of these meet the requirements for a Technical Proposal (which has yet to be finalized). Two of these (from Wil Baden and Richard Gray) were declared to be comments or advice, and were to be returned to their originators, requesting that they state them as proposals.

One of the proposals, from Roy Martens of Mountain View Press, consisted of the following hand-written note: "7/1/87, Elizabeth — We submit the enclosed document, "Forth Floating Point" by Philip J. Koopman, Jr., MVP-FORTH Series, Volume 3, revised, to The Technical Committee for consideration as the standard for integer and floating point math. It conforms to IEEE Floating Point Standard (Task P754) short. There are no restrictions on its use. Sincerely, Roy Martens". This was attached to a copy of their MVP-FORTH floating-point documentation. My notes fail me here, but I believe it was sent back for rework.

The final proposal (so far) was on the treatment of DO loops with equal arguments (n DUP DO) from Lee Brotzman of Uni-Forth. This was discussed at some length. Lee proposed that such loops should run zero times. Some people felt the proposal should be returned, since it didn't meet our (non-documented) proposal format. Don Colburn felt it should be returned for clarification in the case of +LOOP structures (he thought it didn't work as expected in such cases). Chuck Moore opposed it for historical reasons, stating that DO was intended to always run at least once. Ultimately, the proposal was declared controversial and referred to the DO loop magnet, Ray Duncan, for further analysis and comment.

From the TSC survey, Greg Bailey put together a list of 14 controversial areas. Members of the TSC (except the secretary) were assigned to be magnets for the individual areas. Members of the TSC were to write one paragraph on each of the controversial subareas. (See the accompanying list of magnets and their areas.)

TC Meeting 2

I had to leave for a few hours to do some training for my company's field personnel and missed some of the discussion, but here's what I understand to have occurred:

The TSC unanimously agreed to bring a cell-size proposal (removing all references to 16-bit words) to the TC for ratification. The FIG representative, Bill Ragsdale, invoked the two-week rule, which allows any member to put off a vote for two weeks in order to allow sufficient time for review. This apparently created some discord, and someone moved to disband the TSC. The

TC voted to continue the TSC and everything seemed to be smoothed over by the time I was able to rejoin the proceedings. No one said this was going to be easy!

The Research subcommittee was directed to publish the Forth vendor list on MCI Mail. The Logistics subcommittee was requested to attempt to supply computer and copying facilities at each TC meeting. The Vocabulary Representative was directed to research and resolve any conflicts between ANSI definitions and those used in the Basis document. Elizabeth Rather agreed to clarify the question of guests with CBEMA.

The TC then adjourned to the bar.

Summary and Comments

Well, it's been over a year since the first meeting (October, 1986) organized to develop an ANS Forth standard. Understandably, I think, some of the members feel a fair amount of frustration at the amount of energy being dedicated to administrivia. There was a strong desire to show some small amount of progress. The first such proposal was to send the non-controversial list of words (DUP, SWAP, etc.) to the TC for ratification, but it was pointed out that this was a no-op, since it didn't involve a change to the basis document. Next was an attempt to solve the floored division problem, proposing that division operators with remainders (MOD, /MOD, */MOD) should only work on unsigned values — this was declared controversial. Finally, we had unanimous agreement on the cell-size proposal, but this was tabled due to the invocation of the two-week rule. Even if no one said this would be easy, neither did anyone tell us it would be impossible!

The next meeting is scheduled to take three days, and I think there's a fair chance it will be more productive. Also, we will be returning mail ballots on a couple of key issues: Technical Proposal format, and cell size. There is visible progress from this effort, but it's painfully slow.

The TC directed that members present our current status and request input from both the forthcoming FIG convention and the FORML meetings. Additionally, the research committee was directed to take another shot at gathering vendor input.

Folks, we have a serious effort here to reach out and obtain guidance from the *entire* Forth community. My sense is that the TC is determined to involve as many people as possible, to go the extra mile in gathering input, and to give serious consideration to everyone's points of view. With the way this effort is proceeding, I don't see how anyone has grounds for complaint if they end up with a standard they don't like.

The TC is not a set of finite-state automata. We don't speak with one voice. Some of us (well, me) don't even stay consistent from day to day. By its nature, Forth seems to attract people with a strong sense of individualism and creativity. Still, I sense some general areas of agreement: opening up the process, avoiding attempts to garbage up the language, allowing for future extensions, and proceeding quickly to a draft standard we can stand behind.

If you have a strong opinion, I encourage you to get involved and join the TC. If you have a proposal, write it up and send it to the TC secretary. If you have some unformed thoughts on a subject, get in touch with one of the TC members and ask

for help on developing a proposal. The greatest problem I sense is the general apathy of the Forth community. There are no automatics; you may think that something is so obvious that it will certainly be changed, but that probably won't happen unless someone is there to champion the cause. If your Forth vendor is not listed among the TC membership, perhaps you should question them about it. If you are a Forth vendor, it's to your advantage to attempt to sway the committee to do things in such a way that your products can be stamped "ANSI Forth" with a minimum of changes to your product.

In case there's any doubt, this is not an official communique from the ANS Forth Technical Committee, only the notes of a non-finite-state automaton. My thanks to Ray Duncan for some helpful reminders about the meeting activity.

Jerry Shifrin works for MCI, and is well known for his excellent East Coast Forth Board (703-442-8695).

List of Topics and "Magnets"

<u>Topic</u>	<u>Magnet</u>
Vocabularies and :	John Stevenson
Mass Storage (blocks and text)	Andy Kobizar
Loops, EXIT, and Termination	Ray Duncan
Division	Bob Smith
Documentation	Gary Betts
Testing	Chuck Moore
Assemblers	Greg Bailey
Controlled Reference Set	Mike Nemeth
ROMability	Martin Tracy
Host and File Structures	Don Colburn
Interpreter	Dean Sandersen
\, >BODY, [\], etc.	Charles Keane
Numeric output	John Gotwals
Control, ABORT, QUIT, etc.	Wil Baden

A 6502 ASSEMBLER

CHESTER H. PAGE - SILVER SPRING, MARYLAND

The only weakness I have encountered in Forth is the unavailability of primitive subroutines, called by JSR. For example, I have written a floating-point routine which has fast multiplication and division (using primitives), but slow SIN and similar functions. The polynomial approximation to SIN is slowed by high-level looping and iteration. If the computation routine for SIN were written as a primitive, using JSR to call repeated multiplications as object code subroutines, it would be faster.

To correct this defect, I have written a Forth 6502 assembler designed, not for long programs, but for assembling primitive words, one by one. It has provision for 20 labels and 20 label references (per word); this should be sufficient, but it can easily be increased. This assembler is not computer specific; it requires only that the computer is 6502 based.

Since the 6502 uses 100 - 1FF hex as its stack area (return stack), addresses in this range never appear as target addresses in source code. I take advantage of this by using 101, 102, 103, etc. as the label names, and use the low byte to index into an array of label addresses. During the first pass of the assembly, label targets are compiled as label names (single-word addresses), to be replaced by single-word label addresses in the second pass. In the case of a branching instruction, where only one byte is to be compiled, the low byte of the label name is compiled. On the second pass, this byte will be replaced by the jump offset. In both cases, the compilation address of the target is saved in a reference table, along with a flag to distinguish between branching

commands and commands requiring an absolute address. When a label assignment is encountered, its location address is saved in an array of label addresses, indexed by the low byte of the label name. In the second pass, the reference table is stepped through, a label name is found at a compilation address and is converted to the corresponding label address, or the offset for a branch is computed and compiled.

Why shouldn't extensions to the nucleus use primitives?

The 6502 mnemonic words are in Ragsdale's form, consisting of the standard mnemonics suffixed by commas (to distinguish, for example, between ADC as a hex number and ADC, as a mnemonic). Each mnemonic word returns a base opcode single-byte value and a single-precision number which serves as an admissibility key for rejection of inadmissible addressing modes. Roughly speaking, each addressing mode adds a characteristic number to the base value of each opcode. There are several exceptions; these are identified by special bits in the admissibility keys.

Absolute address modes are distinguished from zero-page address modes by examining the address given with a command. X and Y are, thus, defaulted to zero-page modes, but upmoded when addresses above 200 are given ("addresses" between 100 and 200 are label names).

Each mode is assigned a single-bit number for logical comparison with admissibility keys. There are two exceptional cases; the Y mode and the immediate mode (#) have additional bits. These are used to allow zero-page Y to be used with LDX, and STX, and to change the opcode increment produced by # in the cases of LDX, LDY, CPX, and CPY.

Screen #3 gives the mode identifications, keys, and effects on opcodes. Screen #9 gives JSR, and the special words C, and C, which provide for compiling data with address labels; also the time and space savers ^ and GONEXT. Screen #11 gives illustrative examples.

JSRs within a word are handled by labels; JSRs to other words (primitives ending in RTS) are supplied with target addresses by

```
` <name> >BODY JSR,
```

or

```
^ <name>
```

JSRs to entry points in ROM are supplied with addresses, e.g., in the Apple, FC58 JSR, to clear the screen.

Test Results

My 13-digit-precision, floating-point system required 180 milliseconds to compute FSIN. Simply replacing all stack manipulations with primitives reduced this time to 100 ms. Replacing the high-level loop evaluation of the polynomial approximation with a primitive (collection of

```

ASSEMBLER SCR # 1
0 \ Assembly sample
1 \ Conventional format
2 \ LDA #0
3 \ LDY ##80
4 \ L1 STA 300,Y
5 \ DEY
6 \ BPL L1
7 \ JMP NEXT
8
9 \ Format for this assembler
10 \ ASSEMBLE TEST
11 \ 0 # LDA, 80 # LDY, 101 300 ,Y STA, DEY, 101 BPL, GONEXT
12 \ END
13
14 -->
15

```

27JUN87CHP

```

ASSEMBLER SCR # 2
0 \
1 HEX
2 VOCABULARY ASSEMBLER
3 ASSEMBLER DEFINITIONS
4 VARIABLE MODE
5 VARIABLE MODE.KEY
6 14 ARRAY LABEL.TABLE \ Provide for 20 labels, and
7 CREATE REF.TABLE 0 , 0 , 38 ALLOT \ for 20 targets
8 VARIABLE REF.POINTER
9 -->
10
11
12
13
14
15

```

29JUN87CHP

```

ASSEMBLER SCR # 3
0 \ Modes
1 : ZP 0 MODE ! 0 MODE.KEY ! ; \ Adds 4 to opcode
2 : ,X 1 MODE ! 1 MODE.KEY ! ; \ Adds 14 (zero page,X)
3 : ,Y 2 MODE ! 202 MODE.KEY ! ; \ Adds 14 - LDX, STX, only
4 : X) 3 MODE ! 4 MODE.KEY ! ; \ Adds 0 (ZP,X)
5 : )Y 4 MODE ! 8 MODE.KEY ! ; \ Adds 10 (ZP),Y
6 : # 5 MODE ! 110 MODE.KEY ! ; \ Adds 8 Immediate
7 : ,A 6 MODE ! 20 MODE.KEY ! ; \ Adds 8 Accumulator
8 : ) 7 MODE ! 40 MODE.KEY ! ; \ Adds 20 - Indirect JMPs only
9 \ 8 Adds C - Absolute address
10 \ 9 Adds 1C - Absolute,X
11 \ A Adds 18 - Absolute,Y
12
13 CREATE ADD.TABLE \ Indexed by mode value
14 1404 , 0014 , 0810 , 2C08 , 1C0C , 18 C ,
15 -->

```

19JUN87CHP

```

ASSEMBLER SCR # 4
0 \ A is a given address
1 \ C is address returned by opcode mnemonic
2 : RANGE.CHECK ( A C---A C) OVER 100 U< 0= \ True, absolute addr
3   IF MODE @ DUP 3 < IF DROP 8 MODE +! \ modify ZP modes
4     MODE.KEY @ 202 = IF 200 MODE.KEY ! THEN \ Absolute,Y
5     ELSE 5 < ABORT" Illegal Opcode" THEN THEN ;
6
7 : CODE.CHECK ( C---C)
8   DUP 1+ @ \ Admissibility key
9   MODE.KEY @ AND ?DUP \ Test mode
10  IF DUP FF AND ABORT" Illegal Opcode"
11    DUP 100 = IF DROP -2 MODE +! \ For IMMEDIATE and CPX,
12  \ CPY, LDX, LDY, STX, convert to add 0
13    ELSE 200 = 0= ABORT" Code error" \ Not LDX ZP,Y
14    THEN THEN ;
15 -->

```

```

ASSEMBLER SCR # 5
0 \
1 : LABEL.SAVE FF AND DUP LABEL.TABLE @ \ Not new label?
2   ABORT" Duplicate label"
3   HERE SWAP LABEL.TABLE ! ; \ Save label address
4
5 : LC1 SP@ S0 4 - = IF SWAP LABEL.SAVE THEN ;
6 : LC2 SP@ S0 6 - = IF ROT LABEL.SAVE THEN ;
7
8 : COMPILE.ADDRESS ( A---)
9   DUP FF00 AND DUP 100 = \ Is it a label?
10  IF HERE REF.POINTER @ 0 OVER C! \ Full address label needed
11    1+ ! \ Save compilation address
12    3 REF.POINTER +! \ Advance for next entry
13  THEN
14  IF , ELSE C, THEN ; \ Compile absolute address or ZP byte
15 -->

```

```

ASSEMBLER SCR # 6
0 \ CREATE operators for defining mnemonics
1 \ Multimode opcodes
2 : M/CPU CREATE 2 ALLOT C, , DOES> LC2 RANGE.CHECK
3   CODE.CHECK
4   C@ MODE C@ ADD.TABLE + C@ + C, \ Adjust opcode
5   COMPILE.ADDRESS ZP ;
6
7 \ Single-mode opcodes
8 : CPU CREATE 2 ALLOT C, DOES> LC1 C@ C, ZP ;
9
10 : BRANCHES CREATE 2 ALLOT C, DOES> LC2
11   C@ C, C,
12   HERE 1- REF.POINTER @ 1 OVER C! \ Branch offset needed
13   1+ ! \ Save compilation address
14   3 REF.POINTER +! ZP ; \ Advance for next entry
15 -->

```

```

ASSEMBLER SCR # 7
0 \ Second pass replaces stored label targets          19JUN87CHP
1 : SECOND.PASS
2   BEGIN -3 REF.POINTER +! REF.POINTER @ DUP 1+ @
3     \ Find label compilation address
4   DUP WHILE DUP C@ DUP LABEL.TABLE @ \ Label address
5     3 ROLL C@ \ Word-or-byte flag
6   IF 2 PICK - 1- \ Offset
7     DUP ABS 7F >
8 IF DROP CR ." Branch to " 100 + . ." is too far" SP! QUIT
9   THEN ROT C!
10  ELSE ROT !
11  THEN DROP REPEAT DROP DROP ;
12
13 : CLEAR.TABLES B 1 DO 0 1 LABEL.TABLE ! LOOP
14   REF.TABLE 3 + REF.POINTER ! ;
15 -->

```

```

ASSEMBLER SCR # 8
0 \ Definitions of mnemonics          27JUN87CHP
1 0062 61 M/CPU ADC, 0062 21 M/CPU AND, 0062 C1 M/CPU CMP,
2 0062 41 M/CPU EOR, 0062 01 M/CPU ORA, 0062 E1 M/CPU SBC,
3 0062 81 M/CPU STA, 0062 A1 M/CPU LDA,
4 005E 02 M/CPU ASL, 005E 42 M/CPU LSR,
5 005E 22 M/CPU ROL, 005E 62 M/CPU ROR,
6 007E C2 M/CPU DEC, 007E E2 M/CPU INC,
7 016F E0 M/CPU CPX, 016F C0 M/CPU CPY,
8 036D A2 M/CPU LDX, 016E A0 M/CPU LDY, 017D 82 M/CPU STX,
9 007E 80 M/CPU STY, 007F 20 M/CPU BIT, 003F 40 M/CPU JMP,
10 00 CPU BRK, 18 CPU CLC, D8 CPU CLD, 58 CPU CLI, B8 CPU CLV,
11 CA CPU DEX, 88 CPU DEY, E8 CPU INX, C8 CPU INY, EA CPU NOP,
12 48 CPU PHA, 08 CPU PHP, 68 CPU PLA, 28 CPU PLP, 40 CPU RTI,
13 60 CPU RTS, 38 CPU SEC, F8 CPU SED, 78 CPU SEI, AA CPU TAX,
14 A8 CPU TAY, BA CPU TSX, 8A CPU TXA, 9A CPU TXS, 98 CPU TYA,
15 -->

```

```

ASSEMBLER SCR # 9
0 \ More mnemonics and special definitions          27JUN87CHP
1 90 BRANCHES BCC, B0 BRANCHES BCS, F0 BRANCHES BEQ,
2 30 BRANCHES BMI, D0 BRANCHES BNE, 10 BRANCHES BPL,
3 50 BRANCHES BVC, 70 BRANCHES BVS,
4 : JSR, SP@ S0 4 - = IF SWAP LABEL.SAVE THEN DUP 20 C, ,
5   200 UK IF REF.POINTER @ DUP 0 SWAP C! \ Address is a label
6   1+ HERE 2- SWAP ! \ Save compilation address
7   3 REF.POINTER +! THEN ;
8
9 : ,, SP@ S0 4 - = IF SWAP LABEL.SAVE THEN , ;
10 : C,, SP@ S0 4 - = IF SWAP LABEL.SAVE THEN C, ;
11 : END SECOND.PASS CURRENT @ CONTEXT ! ?EXEC ?CSP ; IMMEDIATE
12 : GONEXT ['] NEXT >BODY JMP, ;
13 : ^ ^ >BODY JSR, ; \ Useful in composite primitives
14 \ e.g., ASSEMBLE PROGRAM ^ A ^ B ^ C GONEXT END
15 -->

```

subroutine jumps) further reduced the time to 74 ms.

These results raise a philosophical question: Does a liberal use of primitives really affect portability? Since the nucleus of Forth depends on primitives, why shouldn't the extension of the nucleus to double-precision or floating-point arithmetic also use primitives? It doesn't seem sensible to saddle floating-point users with slow routines to satisfy an aesthetic pride in colon definitions. Applications using either integer or floating-point arithmetic can be written in completely portable, high-level definitions.

Note on Using the Assembler

The simplest way to use a Forth assembler is to append it to the nucleus dictionary and then assemble a desired application. This leaves the assembler vocabulary in memory between the nucleus and the application. Each compiled application saved to disk includes the entire assembler; this wastes disk space as well as fast-access memory.

This waste can be avoided by a simple routine. When the Forth nucleus is booted, note the name of its last word, and the normal next-word location. Call this address <dp>. (Enter "HERE .") Leave space

for the application by entering "n ALLOT" where n is, say, 5000. Then load the assembler and enter "<dp> DP !" to restore the dictionary pointer to follow the nucleus vocabulary. Assembling the application will locate the application vocabulary as if it were a normal continuation of the nucleus vocabulary. After the application has been assembled, the assembler vocabulary can be eliminated by linking the first word of the application directly to the last word of the nucleus, by entering

```
` <last nucleus name> >NAME
` <first application name>
>LINK !
```

```
ASSEMBLER SCR # 10
0 \ Assembler concluded                                19JUN87CHP
1
2 FORTH DEFINITIONS
3
4 : PRIM -2 ALLOT HERE 2+ , ;
5
6 : ASSEMBLE ?EXEC CREATE ASSEMBLER PRIM
7   [ ASSEMBLER ] CLEAR.TABLES ZP !CSP ;
8
9 IMMEDIATE
10
11 DECIMAL
12
13
14
15
```

```
ASSEMBLER SCR # 11
0 \ SAMPLES                                             27JUN87CHP
1 HEX
2 ASSEMBLE (TEST) 0 # LDA, 80 # LDY, 101 300 ,Y STA, DEY,
3 101 BPL, RTS, END
4 ASSEMBLE TEST ^ (TEST) GONEXT END
5
6 ASSEMBLE TRY 5 STX, 1 # LDX, 101 INX, 7 # CPX, 101 BNE,
7   ^ (TEST) 5 LDX, GONEXT END
8
9 ASSEMBLE HOME FC58 JSR, GONEXT END
10
11 ASSEMBLE THIS 3 # LDY, 101 102 ,Y LDA, CLC, 103 ,Y ADC,
12 102 ,Y STA, DEY, 101 BPL, INY, GONEXT 102 1 C,,
13 2 C, 3 C, 4 C, 103 20 C,, 30 C, 40 C, 50 C, END
14 \ THIS has 2-label entries, and data labelling
15 DECIMAL
```

VECTORED EXECUTION & A FULL-SCREEN EDITOR

RICHARD E. HASKELL - ROCHESTER, MICHIGAN
ANDREW MCKEWAN - LAKEVILLE, MICHIGAN

Vectored execution is a useful tool for directing the flow of control in a program. Various forms of the CASE statement are often used for this purpose. We have found that different types of jump tables are often more convenient, and execute faster, than a corresponding CASE statement. In this paper, we will present three different types of jump tables: a simple jump table containing consecutive execution addresses, a jump table containing both key codes and execution addresses, and a jump table in which the key codes can be computed using any Forth statement. This last form of the jump table will be illustrated by an F83 full-screen editor. Each of the jump tables will be created using a defining word. Both F83 and fig-FORTH versions of these defining words will be given.

A Simple Jump Table

Figure One shows the structure of a jump table called `do.key` that contains the CFAs of five Forth words. The number of entries in the jump table is stored as the first element in the parameter field. When `do.key` is called with a value of 0-4 on the stack, the corresponding word in the jump table will be executed. For example, 2 `do.key` will cause `2word` to be executed. The jump table is created by using the defining word `JUMP.TABLE` in the following form:

```
<#entries> JUMP.TABLE <name>
<0word> <1word> ...
```

For example, the jump table in Figure One is created by the statement

```
5 JUMP.TABLE do.key
0word 1word 2word 3word 4word
```

The colon definition of `JUMP.TABLE` in F83 is

```
: JUMP.TABLE ( -- )
  CREATE DUP , 0 ?DO ` , LOOP
DOES> ( n pfa -- )
  SWAP 1+ SWAP 2DUP @ >
  IF 2DROP
  ELSE SWAP 2* + PERFORM
  THEN ;
```

A powerful method of defining jump tables.

The corresponding definition in fig-FORTH is:

```
: JUMP.TABLE ( -- )
  <BUILDS DUP , 0 DO
  [COMPILE] ` CFA , LOOP
DOES> ( n pfa -- )
  SWAP 1 + SWAP 2DUP @ >
  IF 2DROP
  ELSE SWAP 2 * + @
EXECUTE
  THEN ;
```

An F83 example of using this jump table is shown in Figure Two.

A Jump Table with Arbitrary Stack Values

A limitation of the jump table shown in Figure One is that the key values on the stack that select one of the words to be executed must be consecutive values, starting with zero. This is often easily arranged in small, dedicated systems where one scans only a few keys. With a standard keyboard, on the other hand, one usually has available the ASCII code of the key that has been pressed. In this case, it is more convenient to have a jump table that contains both the key (ASCII) code and the CFA of the word to be executed when that key code is on the stack. The jump table shown in Figure Three is such a table. In this case, the first element in the table is the number of key code/CFA pairs, not counting the final, default CFA (`CHROUT`). This jump table is created using the defining word `MAKE.TABLE` as follows:

```
MAKE.TABLE do.key
0 FKEY
8 BKSPACE
81 QWORD
27 ESC
-1 CHROUT
```

The colon definition of `MAKE.TABLE` in F83 is:

```
: MAKE.TABLE ( -- )
  CREATE HERE 0 , 0
  BEGIN BL WORD NUMBER DROP
  DUP 1+ WHILE , ` , 1+
  REPEAT DROP ` , SWAP !
DOES> ( n pfa -- )
```



```
DUP 2+ SWAP @ 0
DO 2DUP @ =
  IF NIP 2+ LEAVE
  THEN 4 + LOOP
PERFORM ;
```

The corresponding definition in fig-FORTH is:

```
: MAKE.TABLE ( -- )
<BUILDS HERE 0 , 0
  BEGIN BL WORD NUMBER DROP
  DUP 1+
  WHILE , [COMPILE]
  ` CFA , 1 +
  REPEAT DROP ` CFA ,
  SWAP !
DOES> ( n pfa -- )
  DUP 2 + SWAP @ 0
  DO 2DUP @ =
  IF SWAP DROP 2 + LEAVE
  ELSE 4 +
  THEN LOOP
@ EXECUTE ;
```

An example of using this jump table is shown in Figure Four.

Creating the Jump Table with Forth Words

The jump table given in Figure Three is a convenient, generalized jump table. However, you need to know the numerical value of each key code before creating the jump table with MAKE.TABLE. It would be even more convenient if you could use Forth statements to compute the key codes during the creation of the jump table. We will use a new defining word called EXEC.TABLE to define this same jump table. The jump table shown in Figure Three is created by executing the following statements:

```
EXEC.TABLE do.key
  0 |FKEY (functionkeys)
CONTROL H | BKSPACE
          (backspace key)
ASCII Q | QWORD (key Q)
H' 2B | ESC (quit to DOS)
```

In this method of constructing the jump table, the statements to the left of the vertical bar | can be any Forth statement that leaves a numerical value on the stack. The

vertical bar itself is a Forth word whose colon definition is:

```
: | ( addr n -- addr )
, ` , 1 OVER +! ;
```

The corresponding fig-FORTH definition is:

```
: | ( addr n -- addr )
, [COMPILE] `
CFA , 1 OVER +! ;
```

This word commas into the jump table the key code value on the stack as well as the CFA of the Forth word following |. It also adds 1 to the pair count stored in the first element of the jump table.

The colon definitions of EXEC.TABLE and DEFAULT: in F83 are:

```
: EXEC.TABLE ( -- )
CREATE HERE 0 ,
DOES> ( n pfa -- )
  DUP 2+ SWAP @ 0
  DO 2DUP @ =
  IF NIP 2+ LEAVE
  THEN 4 + LOOP
PERFORM ;
```

and

```
: DEFAULT: ( addr -- )
DROP ` , ;
```

The corresponding definitions in fig-FORTH are:

```
: EXEC.TABLE ( -- )
<BUILDS HERE 0 ,
DOES> ( n pfa -- )
  DUP 2 + SWAP @ 0
  DO 2DUP @ =
  IF SWAP DROP 2 + LEAVE
  ELSE 4 + THEN LOOP
@ EXECUTE ;
```

and

```
: DEFAULT: ( addr -- )
DROP [COMPILE] ` CFA , ;
```

Note that DEFAULT: stores the CFA of the default word in the jump table, after dropping the address of the first element in

these words call built-in words rather than use BIOS or DOS calls. We can, therefore, redisplay the entire screen without any noticeable delay. If BIOS or DOS calls are used for EMIT and TYPE, the editor should be modified to update only changed parts of the screen, in order to obtain an acceptable response time.

In summary, the EXEC-TABLE defining word given in screen 7 of Figure Five provides a powerful method of defining jump tables. Two examples of such jump tables are given in screens 8 and 9. It is, obviously, a simple matter to modify this full-screen editor by adding and subtracting entries to these jump tables.

	do.key	
pfa	5	
	0word	0
	1word	1
	2word	2
	3word	3
	4word	4

Figure One. Structure of CFA-only jump table.

	do.key	
pfa	4	
	0	
	FKEY	
	8	
	BKSPACE	
	81	
	QWORD	
	27	
	ESC	
	CHROUT	

Figure Three. Structure of key code-CFA jump table.

the jump table (left on the stack by HERE when the jump table was created). As an example of using EXEC.TABLE, a full-screen editor that can be added to F83 will be described in the next section.

F83 Full-Screen Editor

Figure Five gives the listing of a full-screen editor we have added to the IBM PC version of F83. This full-screen editor is activated by typing V from the editor. For example, to edit screen 26, you would type:

```
26 EDIT
V
```

Pressing the ESC key from within the full-screen editor returns you to the F83 editor. Typing DONE will then save any changes you have made, and will return you to Forth.

Note that the definition of V in screen 9 is simply:

```
: V ( -- )
  .MODE BEGIN EDIT-AT
  KEY DO-KEY AGAIN ;
```

After waiting for each key to be pressed, the jump table DO-KEY directs control to the proper Forth word to be executed. The jump table DO-KEY defined in screen 9 handles all control characters used, and defaults to VCHAR, which either inserts or overwrites a character at the current cursor position. On the IBM PC, the function keys and the cursor keys on the numeric keypad return an ASCII code of 0 when called by KEY. In this case, KEY must be called again to retrieve the scan codes for these keys. Note that in the DO-KEY jump table, this is handled by the word FKEY. The colon definition of FKEY is simply:

```
: FKEY KEY DO-FKEY ;
```

where DO-FKEY is another jump table defined by EXEC-TABLE in screen 8. This jump table handles all the function and cursors keys, PgUp, PgDn, Ins, and Del, etc.

The shadow screens in Figure Five describe the functions of the words corresponding to the various keys. Many of

```
Scr # 33      A:CSE241.BLK
0 \ Figure 2
1
2 : jump.table ( -- )
3   CREATE DUP , 0 ?DO ' , LOOP
4   DOES> ( n pfa -- )
5     SWAP 1+ SWAP 2DUP @ > IF 2DROP
6     ELSE SWAP 2* + PERFORM THEN ;
7
8 : 0word CR ." This is word 0 " ;
9 : 1word CR ." This is word 1 " ;
10 : 2word CR ." This is word 2 " ;
11 : 3word CR ." This is word 3 " ;
12 : 4word CR ." This is word 4 " ;
13
14 5 jump.table jump.test 0word 1word 2word 3word 4word
15
ok

3 jump.test
This is word 3 ok
0 jump.test
This is word 0 ok
2 jump.test
This is word 2 ok
4 jump.test
This is word 4 ok
1 jump.test
This is word 1 ok
5 jump.test ok
```

Figure Two. Example use of jump table do . key.

```
Scr # 31      A:CSE241.BLK
0 \ Figure 4a
1
2 : make.table ( -- )
3   CREATE HERE 0 , 0
4   BEGIN BL WORD NUMBER DROP
5   DUP 1+ WHILE , ' , 1+
6   REPEAT DROP ' , SWAP !
7   DOES> ( n pfa -- )
8     DUP 2+ SWAP @ 0
9     DO 2DUP @ = IF NIP 2+ LEAVE
10    THEN 4 + LOOP
11   PERFORM ;
12
13
14
15
ok

32 list
Scr # 32      A:CSE241.BLK
0 ( Figure 4
1 )
2 : default.word ." This is the default word " . ;
3 : 3word ." This is word 3" ;
4 : 5word ." This is word 5" ;
5 : 17word ." This is word 17" ;
6
7 make.table make.test
8     3 3word
9     5 5word
10    17 17word
11    -1 default.word
12
13
14
15
ok

5 make.test This is word 5 ok
3 make.test This is word 3 ok
17 make.test This is word 17 ok
14 make.test This is the default word 14 ok
```

Figure Four. Example use of key code-CFA jump table.

Figure Five. Listing of F83 full-screen editor for IBM PC.

```

1
0 \ Editor Visual Mode Load Screen          03Mar87AM \ Editor Visual Mode Load Screen          03Mar87AM
1 EDITOR ALSO DEFINITIONS
2 2 9 THRU
3 ONLY FORTH ALSO DEFINITIONS
4 CR .( Visual Editor Loaded )
5
6
7
8
9
10
11
12
13
14
15

2
0 \ Display                                03Mar87AM \ Display                                03Mar87AM
1 VARIABLE INSERTING                        INSERTING Current insert mode.
2 : .MODE ( -- )                            .MODE Display current mode.
3 45 0 AT INSERTING @                       INS Toggle between insert and overwrite mode.
4 IF ." Insert " ELSE ." Overwrite" THEN ; (REDO-LINE) Redisplay a line.
5 : INS ( -- ) INSERTING @ NOT INSERTING ! .MODE ; (REDO-SCREEN) Redisplay entire screen.
6                                             REDO-LINE Redisplay current line and mark screen as modified.
7 : (REDO-LINE) ( line# -- )                REDO-SCREEN Redisplay screen and mark as modified.
8 DX OVER DY + AT C/L * 'START + C/L TYPE ;
9 : (REDO-SCREEN) ( -- )
10 DX 6 + 0 AT SCR ?
11 L/SCR 0 DO [ FORTH ] I (REDO-LINE) LOOP ;
12
13 : REDO-LINE ( -- ) LINE# (REDO-LINE) MODIFIED ;
14 : REDO-SCREEN ( -- ) (REDO-SCREEN) MODIFIED ;
15

3
0 \ Character Insert/Delete                13Apr87AM \ Character Insert/Delete                04Mar87AM
1 : (VCHAR) ( char -- )                    (VCHAR) Insert or overwrite a character at the cursor.
2 INSERTING @ IF 'CURSOR DUP 1+ #AFTER 1- CMOVE> THEN VCHAR Ignore non-printable characters.
3 'CURSOR C! ( store char ) REDO-LINE 1 C ( move cursor ) ; DEL Delete the character at the cursor.
4 : VCHAR ( char -- )                      BKSP Backspace and erase a character.
5 DUP BL - 95 U< IF ( printable ) (VCHAR) ELSE DROP THEN ;
6 : DEL ( -- )
7 'CURSOR #AFTER 1 DELETE ( delete 1 char ) REDO-LINE ;
8 : BKSP ( -- )
9 COL# IF -1 C ( move cursor back ) INSERTING @ IF DEL
10 ELSE BL 'CURSOR C! REDO-LINE THEN THEN ;
11 : TAB ( -- )
12 4 COL# 3 AND - ( # positions to next tab stop )
13 INSERTING @ IF " " DROP OVER 'CURSOR #AFTER INSERT
14 REDO-LINE THEN C ( move cursor to tab stop ) ;
15

```

4

```

0 \ Cursor Movement
1 : RIGHT 1 C ;
2 : LEFT -1 C ;
3 : UP C/L NEGATE C ;
4 : DOWN C/L C ;
5 : CRR 1 +T ;
6 : EOL LINE# T 'LINE C/L -TRAILING NIP C ;
7 : HOME TOP ;
8 : END TOP 'START C/SCR -TRAILING NIP C ;
9 : PGUP SCR @ ( don't go past screen 0 )
10 IF ?STAMP B (REDO-SCREEN) ELSE BEEP THEN ;
11 : PGDN SCR @ CAPACITY 1- < ( don't fall off end )
12 IF ?STAMP N (REDO-SCREEN) ELSE BEEP THEN ;
13 : ALT ( -- : alternate between screen and shadow )
14 ?STAMP A (REDO-SCREEN) ;
15

```

14

```

17Apr87AM \ Cursor Movement
RIGHT Move cursor right.
LEFT Move cursor left.
UP Move cursor up one line.
DOWN Move cursor down one line.
CRR Move cursor to beginning of next line.
EOL Move cursor to end of current line.
HOME Move cursor to top of screen.
END Move cursor to end of screen.
TAB Move cursor to next tab position.
PGUP Go to previous screen.
PGDN Go to next screen.
ALT Alternate between a screen and its shadow.

```

03Mar87AM

5

```

0 \ Line Insert/Delete
1 : NEWL ( -- )
2 'LINE C/L OVER #END INSERT 'LINE C/L BLANK REDO-SCREEN ;
3 : DELL X REDO-SCREEN ;
4 : DEOL 'C#A BLANK REDO-LINE ;
5 : REPL 'INSERT 1+ 'LINE C/L CMOVE REDO-LINE ;
6 : INSL NEWL REPL ;
7 : SAVL KEEP 0 18 AT .LINE ;
8
9 : VWIPE WIPE (REDO-SCREEN) ;
10 : VDISCARD DISCARD CHANGED OFF (REDO-SCREEN) ;
11
12 : VSPLIT ( -- ) SPLIT REDO-SCREEN ;
13 : VJOIN ( -- ) JOIN REDO-SCREEN ;
14
15

```

04Mar87AM \ Line Insert/Delete

```

NEWL Insert a blank line before the current line.
DELL Delete the current line. Save in insert buffer.
DEOL Delete to the end of the line.
REPL Replace line from the insert buffer.
INSL Insert contents of insert buffer before current line.
SAVL Copy line to insert buffer.
VWIPE Wipe screen clean.
VDISCARD Ignore changes made to screen.
VSPLIT Split line at cursor.
VJOIN Join next line at cursor.

```

04Mar87AM

6

```

0 \ Other Visual Operations
1 : DELW ( -- )
2 'C#A 2DUP TUCK BL SCAN BL SKIP NIP - DELETE REDO-LINE ;
3 : +WORD ( -- )
4 'CURSOR #REMAINING TUCK BL SCAN BL SKIP NIP - C ;
5
6 : (-WORD) ( addr1 len1 -- adr2 len2 )
7 DUP 0 ?DO 2DUP + 1- C@ BL = ?LEAVE 1- LOOP ;
8 : -WORD ( -- )
9 'CURSOR C@ BL <> IF -1 C THEN
10 'START CURSOR -TRAILING (-WORD) R# ! DROP ;
11
12
13
14
15

```

03Mar87AM \ Other Visual Operations

```

DELW Delete the next word.
+WORD Move cursor to the beginning of the next word.
(-WORD) Remove non-blank characters from a string.
-WORD Move cursor the beginning of previous word.

```

03Mar87AM

```

7
0 \ Execution Table
1 : EXEC-TABLE ( -- addr )
2 CREATE HERE 0 ,
3 DOES) ( n {pfa} -- )
4 DUP 2+ SWAP @ 0
5 ?DO 2DUP @ = IF NIP 2+ LEAVE THEN 4 + LOOP
6 PERFORM ;
7
8 : ( addr n -- addr ) , ' , 1 OVER +! ;
9
10 : DEFAULT: ( addr -- ) DROP ' , ;
11
12
13
14
15

```

```

17
04Mar87AM \ Execution Table
04Mar87AM
EXEC-TABLE Define an execution table. Items are compile with
!. At runtime a value is placed on the stack and the table
is searched for a matching value. If it is found, the
selection value is dropped and the corresponding word is
executed. Otherwise the default word is executed with the
selection value still on the stack.

! Compile a table entry.
DEFAULT: Compile the default action and end table definition.
If no default action is desired use DEFAULT: DROP.

```

```

8
0 \ IBM Function keys
1 EXEC-TABLE DO-FKEY
2 59 : NOOP ( F1 ) 60 : ALT ( F2 )
3 61 : SAVL ( F3 ) 62 : DELL ( F4 )
4 63 : REFL ( F5 ) 64 : INSL ( F6 )
5 65 : NOOP ( F7 ) 66 : NOOP ( F8 )
6 67 : VWIPE ( F9 ) 68 : VDISCARD ( F10 )
7
8 71 : HOME ( Home ) 72 : UP ( Up )
9 73 : PGUP ( PgUp ) 75 : LEFT ( Left )
10 77 : RIGHT ( Right ) 79 : END ( End )
11 80 : DOWN ( Down ) 81 : PGDN ( PgDn )
12 82 : INS ( Ins ) 83 : DEL ( Del )
13 115 : -WORD ( ^Left ) 116 : +WORD ( ^Right )
14 DEFAULT: DROP
15

```

```

18
04Mar87AM \ IBM Function keys
03Mar87AM
DO-FKEY Table of actions for IBM function and special purpose
keys. The scan code is on the stack.

```

```

9
0 \ Editor Visual Mode
1 : FKEY ( -- ) KEY DO-FKEY ;
2 : ESCAPE ( -- )
3 ?STAMP 0 (REDO-LINE) 'START 'VIDEO B/BUF CMOVE QUIT ;
4 EXEC-TABLE DO-KEY
5 0 : FKEY CONTROL E : EOL
6 CONTROL H : BKSP CONTROL I : TAB
7 CONTROL J : VJOIN CONTROL M : CRR
8 CONTROL N : NEWL CONTROL S : VSPLIT
9 CONTROL T : DELW CONTROL U : DEOL
10 CONTROL Y : DELL 27 : ESCAPE
11 DEFAULT: VCHAR
12
13 : V ( -- )
14 .MODE BEGIN EDIT-AT KEY DO-KEY AGAIN ;
15

```

```

19
04Mar87AM \ Editor Visual Mode
03Mar87AM
FKEY Get exented key code and perform function key action.
ESCAPE Exit from visual mode to editor command mode.
DO-KEY Execution table for control keys.
V Enter visual mode from the editor.

```

Figure Six. F83 partial editor glossary.

#AFTER	(-- n)	# chars after cursor on current line
#END	(-- n)	# chars between line start and screen end
#REMAINING	(-- n)	# chars after cursor on screen
'C#A	(-- addr n)	address of cursor and #AFTER
'CURSOR	(-- addr)	address of char at cursor
'INSERT	(-- addr)	address of insert buffer
'LINE	(-- addr)	address of beginning of current line
'START	(-- addr)	address of start of screen
+T	(n --)	go to beginning of line relative to current
?STAMP	(--)	update screen id if changed
A	(--)	toggle screen and its shadow
AT	(col row --)	position cursor
B	(--)	go back one screen
C	(n --)	move cursor n characters right or left
C/L	(-- n)	characters per line (64)
CAPACITY	(-- n)	number of blocks in the file
COL#	(-- n)	current column number
DELETE	(addr len # --)	delete # characters from string
DISCARD	(--)	ignore latest changes
DX	(-- n)	column for upper left corner of screen
DY	(-- row)	row for upper left corner of screen
INSERT	(a1 n1 a2 n2 --)	insert string 1 at beginning of string 2
JOIN	(--)	copy next line at cursor
KEEP	(--)	put current line in insert buffer
L/SCR	(-- n)	lines per screen (16)
LINE#	(-- n)	current line number
MODIFIED	(--)	mark screen as modified
N	(--)	go to next screen
R#	(-- addr)	variable for cursor position (0-1023)
SPLIT	(--)	split line at cursor
T	(n --)	go to beginning of line n
TOP	(--)	go to top of screen
WIPE	(--)	erase screen

PROFILES IN FORTH: John D. Hall

Mike Ham, frequent FD interviewer, recently spoke with one of the Forth Interest Group's Board of Directors. John David Hall shared his professional background, and his views of FIG and of the greater Forth community.

MH: How long have you been FIG chapter coordinator?

JH: Five years; I started in 1983. I went to a board meeting, and there were two things going on. I was interested in the chapter mailings: John Cassady was sending the handouts to the various chapters. At the board meeting, the suggestion came up that maybe the work that John was doing wasn't getting completed soon enough. I volunteered to help John, and the board said, "Great, John James really needs some help."

I said, "John James?" And they said, "Yes, he's trying to put together some chapters." They wanted to formalize the process and set up a formal contact between FIG and the chapters. John James had already started writing up the documents, and he also needed some help. And that's how I got started with the chapters.

MH: How many chapters did FIG have then?

JH: About 16 informal chapters. Many are the core of what we have now as chapters. We had contacts in Tokyo already, the New York chapter, the Potomac chapter, the

Silicon Valley chapter, the Orange County chapter.

MH: How many do we have now?

JH: We have a list of 80 chapters. Last year, I sent out recertification forms and got them back from about 50, but out of the remaining 30 there are chapters that exist and are very strong, like the Potomac chapter, who never answer any inquiries. Lately I call people and ask if there really is a chapter there. I thought the Chicago chapter had

Forth is used more and more, but it is somewhere underneath..."

fallen apart. But today I got a contact with one of the people, and it's very active; I just had the wrong person. The person who had been the contact had moved on and left it with someone who had not gotten ahold of us yet.

MH: The chapter movement is important for remote members.

JH: We wanted to start the chapters as a way of keeping the membership growing. You really can't do it from a central place. We can do only so much with *Forth Dimensions*. The authority and the ability to gather new members needs to be passed on to the chapters.

MH: What do successful chapters have in

common? Can you tell what makes a chapter work?

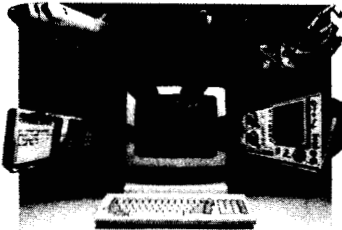
JH: Yes: dynamic personality, of the leader or leaders. We're really fortunate in Silicon Valley because we have several dynamic people. If one person drops out, somebody else steps in. I've tried to encourage other chapters to start — to see if they can't find these dynamic people to help them run the chapter. When they haven't been able to find such people, the chapter doesn't seem to be able to survive. It takes somebody who's dedicated to helping other people and who's really enthusiastic about it.

What we're beginning to find out is that it takes more than one person. One person is important. But as soon as that person gets tired, who do you pass it on to? It really takes two, or three. You really need to find a core of two or three — you need to find a team, so that when one person's on vacation, things don't fall apart. When one person wants to go to Thanksgiving, or to FORML, things can't fall apart. So you need a team of people. And that's really what it takes.

MH: You mentioned in the Silicon Valley chapter meetings a morning session and afternoon session. How do those work?

JH: When the Silicon Valley chapter first started, the morning session was called FORML, just like the FORML conference. The reason was, we had technical subjects in the morning. In the afternoon, we had people showing applications, things they had done. It's become a little confused

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



Find Out How To Implement
Real-Time Systems In:

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

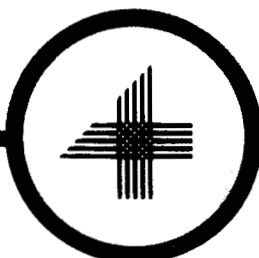
... on time and under budget.

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

**(213) 372-8493,
Ext. 444.**

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

ON TIME. UNDER BUDGET.



FORTH, Inc.

lately, because we have technical papers in the afternoon, as well as in the morning, or we have applications in the morning as well as the afternoon. So the structure of morning and afternoon has been blurred. I'm not quite sure why that is. We also see a decline in the number of papers being presented. In the Silicon Valley we see more applications being demonstrated than code being demonstrated. I think it's just our lack of emphasis on trying to get the technical people there.

MH: What do other chapters do?

JH: San Diego County has an interesting format. They meet at lunchtime. Silicon Valley meets the fourth Saturday of every month, and it's pretty much an all day: 10 o'clock in the morning until 5 o'clock. San Diego meets every Thursday at lunchtime, and it's very informal. I was talking to them this morning. Guy Kelley is more or less the leader of the group, and it's a roundtable. Physically, there's a U-shaped table. Guy sits in the middle of the U, and whoever wants to talk gets up and goes up to the blackboard and the podium and starts talking. They don't have a secretary or treasurer, and they have been meeting successfully for five or six years that way. Everybody brings his lunch. They have a good group; they're kind of a rebellious group. If someone wants to get up and say something controversial, they do. And often they get into a good discussion.

MH: What about your own background? When did you first find Forth?

JH: I was a physics and chemistry major in Alaska and at Berkeley, and I didn't quite finish college when I went into the Army. I came back from the Army in 1963 and went back to Berkeley. By then I wasn't as interested in chemistry as I had been earlier, but I had to finish my degree, so I went on with a chemistry degree. At the same time, computers — big computers — were beginning to be used, and I learned FORTRAN. I was becoming interested in computers.

I returned to Korea as a civilian, my wife and I were married. I returned to the U.S. in 1967. I decided that computers were what I wanted to be involved in. I found a

chemical company and started working as a lab assistant. They had an IBM 1130. It was a brand-new computer, but it was sitting idle, and I did some programs for the chemical engineers.

In 1970, I worked with an accountant in Oakland, where I now live, using an IBM 1130. I've worked off and on for accountants, and the University of California, in the Agricultural Extension Service, which did a lot of 4H registration. All on the IBM 1130.

MH: What language?

JH: All this was business applications in FORTRAN. When COBOL became available on the IBM 1130, it was in COBOL. This was in 1974. In '75 I read the article about the MITS computer and I bought an Altair computer. I put that together myself. My whole background has always been software; this was the first project where I had picked up a soldering iron and worked with hardware.

That got me into microcomputers. I was still working at the University of California at that time. About two years later the Processor Technology SOL computer came out. I had been working in the accounting service before, and I went there with a proposal: let's put together an accounting package. BASIC was all that was available at that time. So my sister, Beckie Harvey, and I began programming in BASIC and wrote a complete accounting package.

This was around August of 1980, and as we read the Byte articles about Forth we realized this was really the direction we would like to go. Because we were trying to make our BASIC programs as flexible as possible: we were building small tools. I had a sort package I could move from application to application, I had a string data-entry package I could easily move. We were trying to put together these modules, and here was a modular language.

So we started to reprogram everything we had written. We had a general ledger and accounts receivable for convalescent hospitals, vertical markets at that time. We had all these written in BASIC, and they were slow. We started rewriting them in Forth and were well on our way when Processor Technology died on us, right in

the middle. We had an orphan.

We could have continued, and we could have sold the few processors we had. We had about 12 of them installed in several convalescent hospitals, doing accounts receivable. But the Sol was gone. The next question was, where to go? We were stuck.

So in 1981, Beckie and I split up. She went down to Los Angeles and worked for a company, programming in Forth; and I became a consultant and found jobs in Forth. One was for a company called Stafa, where they were beginning to do controllers for ducts: environmental control. They had a controller built right into the ducts, and they could talk to the device through the AC connections. It was a single chip and it had Forth on it.

MH: The 8080?

JH: No, but it was an embedded 8080-like processor; an 8080 controller chip with everything on it, A-to-Ds, everything. A single chip with all the hardware on it.

MH: A long way from the 1130?

JH: A long way from the 1130! 1130s at that time were considered minicomputers, and they were designed as scientific minicomputers, but we had used them as business machines because they were inexpensive.

The Stafa project took about 8 months or so. Then about that time Gary Feierbach had a job to do an engine test device for a company in Ventura. Paul Thomas, Gary Feierbach, Matthew Johnson, and myself were all working as a team. The project was what I now characterize as a typical Forth project: putting out a fire. Somebody had attempted a project, had put a lot of money into it, decided they couldn't do it in the traditional languages, came to us and said, "We really need to get this thing done, and we really need to get it done by a certain date, and we've heard that Forth can do this — do it for us."

Gary was teaching Forth and was doing some hardware at the time, and decided to take on the project.

The project really wasn't well defined. And as we went along, the requirements changed. The client thought we could make the project a little more user-friendly — at

that time the buzzword was "user-friendly." By the end of a year, it turned out we were doing it almost 180 degrees out of phase from the original specifications. Originally, we had started a controller with very little user interface: it was to do engine testing. When we were all done, it was an engineer's workstation, with a much larger user interface.

In most applications, a user interface is a good 50% of the project, so we had almost doubled what we had set out to do in the first place.

MH: Did it come out as a product?

JH: It eventually came out as a product, though it was completed by another group.

After that, I went to work with Rising Star. They had already been working for a year or so on putting together a personal computer, a turnkey kind of computer. You turned it on, it came up in an editor; you hit a calc key and went into the spreadsheet; you hit "mail," you went into the mail programs; you hit the copy key to copy a disk. It was a basic, novice, entry system. That was the concept. At that time, the IBM PC still had not really picked up, and they were doing it on an Epson computer with a Z-80. The interesting part was that we were showing what a Z-80 could really do. It was amazing. We had multiple banks of 64K memory. We put a spreadsheet in one bank, the editor in another bank, the operating system in a third bank, and so on. And graphics — we had graphics that nobody else had at that time.

MH: You actually drew the letters on the screen.

JH: Yes, the editor was What-You-See-Is-What-You-Get; we had bold and italics and all the stuff. And most of the upper-level applications were written in Forth, pretty much in high-level Forth. The operating system was written in assembler, by a company that had started up way back when CP/M was starting. They had written their own workalike CP/M called TP/M. Now they were reforming, with the same people and with their own operating system, CP/M compatible but with extensions. They had one graphics guru; he took care of the graphics engine. We made calls to the

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

HS / FORTH

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 turnkeyed 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 shelled programs) and link to .OBJ 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 (finance) 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 metacompiled tools, or turnkeyed applications royalty free.

HS/FORTH (complete system):	\$395.
HS/FORTH: Tutorial & Ref (500 pg)	\$ 59.
Forth: Text & Reference (500 pg)	\$ 22.
HS/FORTH Glossary	\$ 10.
GIGAFORTH Option (Beta release)	\$245.
(Native Mode from SOFTMILLS, INC.)	

 Visa  Mastercard

HARVARD
SOFTWORKS

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

graphics engine to draw a line or change fonts — that kind of thing. We didn't draw the characters in Forth, we called the graphics engine. But again it was all the Z-80 machine: the same Z-80 machine, with us calling it from different places.

MH: So when you switched the application, you would just drop into the operating system, and it would switch the bank — is that the way it worked?

JH: Yes.

MH: What was your job in the Rising Star project?

JH: My job switched very quickly. Originally, I was to help do the menus. Between applications, there were menus that would allow you to switch applications. The first thing that struck me was that the Forth underneath was very large, very overbuilt. So it was decided to reinstall the underlying Forth. That was a three-person job, and I got involved. We installed a simplified, 83-Standard model with separated heads (like the original F83).

Then we put that underneath the applications — the applications were already building. The most complete was the spreadsheet, so we took that one, moved the new Forth under the spreadsheet, and found that no changes had to be made to the spreadsheet. We had integrated it enough into what the old system looked like, that it really fit underneath — it surprised me. I thought there would be tools that we didn't include, but it didn't happen; it fit in very nicely.

But Rising Star was out on a limb. They were being fed money by Epson, and that's how they were able to keep about 30 programmers going. The programmers were scattered all over the United States. There were a team of 6 or 7 assembly language programmers who did the operating system and the mail program; they were back East. There were a couple of people in Oregon, who did documentation of the team that did the new Forth. I lived in Oakland, Paul Thomas lived in San Francisco, and Ron Braithwaite lived in San Diego county.

This was the first time I had seen programming at a distance. We all worked at home and sent modules back and forth by a

bulletin board system on the team leader's computer. We all used the Epson computer — they were up enough and running so we could use it. We would upload and download modules using the team leader's bulletin board. Every month, everybody would come across the United States to meet in Los Angeles, in Torrance where the headquarters were.

It was a nice environment, pleasant for programmers. It worked as a management tool, and it worked very nicely — particularly for Forth programmers, because we were very independent. We liked to work by ourselves on projects and then come together to pool the pieces and the knowledge, rather than all working together in an office. Really, you do the same thing in an office. You get together, pull off a piece, go work on it, put it back together again. It's just that often you have someone come in and look over your shoulder to see how you're doing, see if you're doing it fast enough. It turns out that if you do it at home, you work more hours on your project than in the office, because you work until ten or twelve at night. At the office you go home at five.

Rising Star ended in December of 1984. I was there for about half the project, about 8 months. By then the IBM was beginning to make an impact. Epson decided they no longer wanted to continue with a Z-80 product, they wanted an 8088 product. They thought we could finish, so they gave us a little extra life. I think they would have cut it off in August or September, but they thought, "Well, we're very close to getting the product done and getting it out the door," so they gave us until December before they cut off the money. We weren't done, and Rising Star just collapsed in December.

If you want to point any fingers, it was because of the editor. The editor was a concept picked up from Forth — it was the typical Forth line editor extended to this glorious What You See Is What You Get. The idea of the line editor is that you edit on the single line in the middle of the screen; as you scroll up or down, the screen moved and you were editing that line. But a line of text extends any distance; you have to mark it — there were a lot of complications. But the whole model was limited in how it was first conceived, and it grew beyond its

bounds.

The original concept should have been modified earlier. Rising Star spent a lot of money putting people in hotels. They took the editor team and locked them in a hotel for three weeks at a time trying to get the product up to a certain level. It was a very intense situation. There was no time even to go back and think. They just plowed along at this point. Probably that should have said the end is coming.

MH: If you get off on the wrong foot with a concept, you can go a long way; but there's a certain point beyond which you need to say, "This was a wrong turn, let's go back to the very beginning and rethink this."

JH: You have to back up, but when money is involved, time is involved, and reputations are involved, sometimes you can't back up. And if you can't back up and you can't go ahead, the fate is sealed. That doesn't seem to be acceptable either, but it often comes to a tragic end.

MH: And you were back to consulting.

JH: I was working for Rising Star as a consultant. From Stafa to Inner Access to Rising Star, I was a consultant. When Rising Star fell, it caught me off guard, and it took me until about March to find some other work. I took a job at Lockheed Research and Development in Palo Alto, and it turned out to be very interesting work. Lockheed's research and development is right next to Stanford, and it gives the area around Lockheed an academic atmosphere. I thought it was an opportunity to get in and do something with Forth that wasn't just grinding out code. Not that all my other jobs were just grinding out code — there were some opportunities to explore. That was in 1985, and now I have worked there almost three years.

MH: Programming in Forth?

JH: Yes. Our major department is called Applied Physics. As a subgroup, we are about 10 people; our sub-group is called "fast processors." The idea is to build fast sensors. The leader of the group is very interested in Forth. Some are software

people, and the rest are hardware people. But in Forth, there is not a clear distinction between the software and hardware people. I have learned a lot more of hardware than I would ever have thought was possible.

Forth has opened my eyes to these things I at one time called black boxes. On the IBM 1130, I knew the language but I didn't know what the operating system did, I didn't know the drivers — everything but the language and application was a black box. There are no black boxes any more. Forth is so simple, I can cut through and see that all these things I called black boxes were probably overly built, complicated structures that didn't really need to be there. Maybe the machinery of the time was more primitive and that may have caused some of complexity.

MH: What processor do you use now?

JH: When I first started at Lockheed, we were using Intel development machines, essentially 8086s and 286s running parallel

processes. They communicated through a common memory on a bus. There were four independent cpu boards communicating through common memory, each board collecting, storing, manipulating, and logging the data.

Our group has now moved to the Novix 4016 chips. We found with the 4010 we can replace hardware with software. If somebody wants a sensor, we can take the 4016, attach it to whatever we want to sense, and run it fast enough that we don't need hardware in between. We don't need much hardware in front of us — maybe an A-to-D. We can do a lot of high-speed sensing.

We were using Computer Cowboy's boards. We have a project now to see if we can't put together a parallel Novix system, where a lot of Novix chips and boards are all working together.

MH: How many?

JH: We're starting with 10 independent cpus connected in parallel. We're begin-

ADVERTISERS INDEX

Bryte - 13
 FORTH, Inc. - 32
 Forth Interest Group - 10, 24, 40
 Harvard Softworks - 33
 Inner Access -36
 Laboratory Microsystems - 8
 Miller Microcomputer Services -12
 Mountain View Press - 6
 Next Generation Systems - 11
 Pair Software - 35
 Silicon Composers - 2

ning to do some of the software. It's a master-and-slaves concept, where the master decides what the whole project is all about and posts on a blackboard jobs to be

Go FORTH™

The ProDOS Forth Language implementation for the Apple Computer //e, //c, //gs and ///

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

Go FORTH is the new FORTH language implementation for the Apple® //e, //c, //gs (//e emulation mode, full //gs version late Fall) and the Apple® ///. It is 100% ProDOS® and SOS® supported. **Go FORTH** code is intercompatible with all **Go FORTH** supported 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 (916) 485-6525
 3201 Murchison Way, Carmichael, California 95608

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

done. The slave's job is to take a task, do it, and post the results back on the blackboard, picking up the next task.

MH: Let's talk a moment about FIG. What is your take on where FIG is going?

JH: FIG, not Forth, has come to a level, a plateau. My impression is that Forth is continuing to be used more and more, but it is somewhere underneath everything. It shows up continuously: Rapid File, VP Planner, the Canon Cat — all the things we point out as examples of where Forth is being used — they're still growing, but FIG is plateaued at the moment.

MH: It sounds like FIG at the beginning was an incubator for the fledgling language and then, once the language was established, FIG looks around for a new direction. I feel as if FIG is still an entry for people to get into Forth, but FIG's mission with regard to established Forth is not clear.

JH: You're right. There wasn't any use of Forth by the common programmer before FIG came along. The only two companies that existed were Forth, Inc. and Miller Microcomputer Services. The FIG group came along and decided that they wanted to build a people's version of Forth. It was a good Forth, and that sparked an interest — now everybody could know about it, everybody could get involved.

Because Forth opens up these black boxes, people who didn't know they could get into compilers, got into compilers. Before, you *used* languages, you didn't inquire as to how they were constructed, why they were there, what they did. Forth was both language and operating system, so you opened up the operating system. You opened a lot of people's eyes, who didn't realize what power was there.

Martin Tracy calls these people hobbyists, but I would say they're really professionals who got into this. They knew hardware, they knew software — they may have

come from other disciplines, but they got into microcomputers at the beginning.

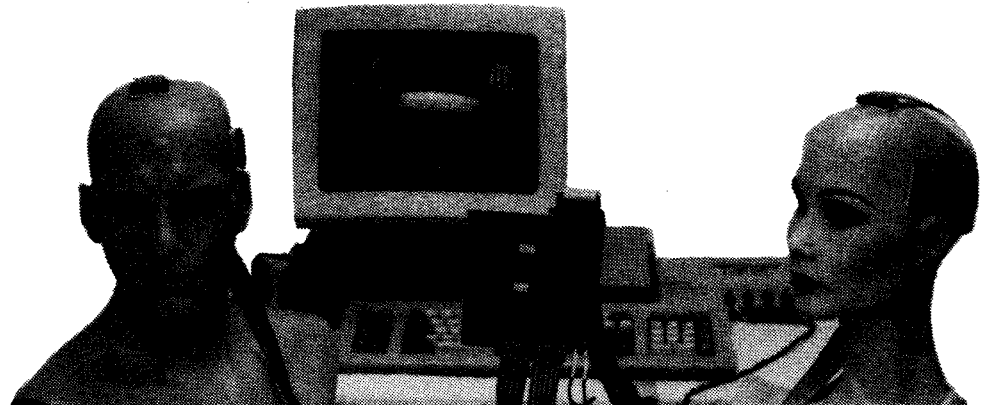
Nowadays, though, you don't go out and buy an Imsai, an Altair, or a Sol and put it together yourself and put your own language into it. You go out and buy a Macintosh, and immediately you have applications sitting in front of you. You may want to go back down and write an application of your own, but you are going *down* to write the application, you're going *down* from the level you were at. You've got to build the user interface, you've got to build a lot of things. So people have different expectations now. Earlier, people had no expectations at all — they had to build everything, and fig-FORTH was a tool to get them through it. That group of people — and they could be called hobbyists — I don't think we can count on as much now.

Fifty percent of FIG are hobbyists, 50% are professionals. We have tried to decide how to support each. We're coming to the conclusion that FIG now satisfies the hob-

GIVEN: Our Forth Super-8 Development System and Your IBM Compatible PC —

YOU CAN:

**Really
Start
Something**



Do your target application right on the target.

Your PC becomes the file server and terminal.

Development board comes complete with development ROM, F83 on PC diskette with terminal emulation and file server software.

Full documentation.

LIST **\$295**



Inner Access Corporation

1155-A Chess Dr., #D, Foster City, CA 94404 (415) 574-8295 Telex 494-3275 INNACC

byists and is the base for the professionals, but it doesn't satisfy the professionals completely. We're looking for techniques to meet those professionals' needs.

MH: And you would be interested to hear from any professionals as to what they need and want.

JH: We'd be interested to hear, but what they need and want is becoming clear. They need money, and they need moral support. They don't want to hear their managers ask, "What's Forth and why are you using it? Why aren't you using C?" They want to hear their managers say, "We've heard about this wonderful idea — you've got to use Forth."

I think we can approach that. This is something we've been discussing at this convention: techniques of helping those people — maybe not directly, but indirectly, by opening management eyes.

MH: What steps do you think FIG needs to take?

JH: Communication, in all senses. In the past, we have left it up to *Forth Dimensions* and the chapters to communicate to our members. Now we have branched out into

GENie. That's another kind of communication. What I'm beginning to see is that there's a whole spectrum of communication: we need a newsletter that may go out monthly, we need chapters talking directly to other chapters (GENie will help that), we need to encourage authors to write general interest articles that can go into magazines that are about applications, but not specifically about Forth.

If I could go back and do things over, I'd start by saying that we won't promote Forth directly. I would take an indirect approach, go around the outside and present some really interesting applications. "Just look at these — aren't these really useful to you? By the way, they're written in Forth. And they're ten man-year kind of projects that took two man-years to do, and the dollars per line of code is x."

I would try that approach. I know many people along the way said that's what we should be doing; but I didn't hear it, and many of the rest of us didn't hear it.

MH: One problem is that when you're working in Forth, it's so evident. It's always hard to talk about the obvious. When you're working in Forth you can tell how productive you're being, you can tell how much you can do with how little. You can tell how

your accumulated tools and understanding add up and give you increasing leverage in a way that I didn't feel I had with Fortran or assembly language. It becomes so obvious it's hard to explain. That's where the idea of the Forth zealot came from: they say it's real good, and when you ask them why, they can't explain — it's hard to explain, because it's so obvious. They finally say, "Just use it." And then you think, "Hmm. Zealot."

JH: One curiosity that has bothered me is, why haven't assembly language programmers used Forth as their language tool? It's obvious that they have the power of writing an assembly language application using Forth and have the interactive ability right in front of them — which they don't have in assembly. They just don't have anything in the assembler along this line. That's a real curiosity to me: why we don't have those people using Forth. A large potential group, but I think the time's past. Those days may have changed; they may be all using C now.

Mike Ham is dp product manager at CTB/McGraw-Hill in Monterey, California.

Rumor Stack by The Inner Interpreter

One of the Big Ten (Five? Seven?) of Forth systems vendors is combing their in-house archives for utilities and/or applications to package as off-the-shelf products. (Whether they give credit, much less provide access, to the underlying Forth remains to be seen.) Did this departure from past practice come from new, middle-managerial blood (a total transfusion, I'm told), or was it the other way around? Of course, people have been saying for years that systems-only vendors can't tackle the market size-and-share problem with just a limited ad budget. You can bet other vendors will be watching the experiment closely.

Meanwhile, a different big hitter appears to be quietly defecting to a less-frustrating market and a less-interesting language (see what I mean). Several programmers I know have been studying those tortuous straits, and now their arguments for Forth are more reasoned than rabid. But the dark side is more powerful than we know, Luke. What's happening here?

A small but well-known Forth shop dropped out of sight a few years ago, but they are back with a proposal in hand for a tasty dollop of the government's budgetary mousse. OK, this is premature, even as rumours go, but it's the kind of project that could put the work of more than one Forth

notable into the sights of the Great Chefs of Washington.

Question of the day: who will offer the first, full-blown Sourceless Forth? Our rangy wrangler, that computer cowboy, mentioned this in his Fireside Chat — and so did some FORML-goers. Good idea — I'd like to try one on for size. Soon, please? (ABEND)

"The Inner Interpreter" listens — send him your juicy tidbits, c/o Forth Dimensions. He's careful, but be advised: the rumors he reports may be no more than rumors.

FIG CHAPTERS

U.S.A.

• ALABAMA

Huntsville FIG Chapter
Tom Konantz (205) 881-6483

• ALASKA

Kodiak Area Chapter
Horace Simmons (907) 486-5049

• ARIZONA

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

• ARKANSAS

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

• CALIFORNIA

Los Angeles Chapter
4th Sat., 10 a.m.
Hawthorne Public Library
12700 S. Grevillea Ave.
Phillip Wasson (213) 649-1428
Monterey/Salinas Chapter
Bud Devins (408) 633-3253
Orange County Chapter
4th Wed., 7 p.m.
Fullerton Savings
Huntington Beach
Noshir Jesung (714) 842-3032
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 Forsberg (305) 252-0108
Tampa Bay Chapter
1st Wed., 7:30 p.m.
Terry McNay (813) 725-1245

• GEORGIA

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

• ILLINOIS

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

• INDIANA

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

• IOWA

Iowa City Chapter
4th Tues.
Engineering Bldg., Rm. 2128
University of Iowa
Robert Benedict (319) 337-7853

Central Iowa FIG Chapter

1st Tues., 7:30 p.m.
Iowa State Univ., 214 Comp. Sci.
Rodrick Eldridge (515) 294-5659
Fairfield FIG Chapter
4th day, 8:15 p.m.
Gurdy Leete (515) 472-7077

• KANSAS

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

• MASSACHUSETTS

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

• MICHIGAN

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

• MINNESOTA

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

• MISSOURI

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

• NEW JERSEY

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

• NEW MEXICO

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

• NEW YORK

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

• NORTH CAROLINA

Raleigh Chapter
Frank Bridges (919) 552-1357

• OHIO

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

• OKLAHOMA

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

• OREGON

Greater Oregon Chapter
Beaverton
2nd Sat., 1 p.m.

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

• **PENNSYLVANIA**

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

• **TENNESSEE**

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

• **TEXAS**

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

• **UTAH**

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

• **VERMONT**

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

• **VIRGINIA**

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

• **WISCONSIN**

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

INTERNATIONAL

• **AUSTRALIA**

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

• **BELGIUM**

Belgium Chapter
4th Wed., 20:00h
Contact Luk Van Look
Lariksdruff 20
2120 Schoten
03/658-6343
Southern Belgium Chapter
Contact Jean-Marc Bertinchamps
Rue N. Monnom, 2
B-6290 Nalines
071/213858

• **CANADA**

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

• **COLOMBIA**

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

• **DENMARK**

Forth Interesse Gruupe
Denmark
Copenhagen
Erik Oestergaard, 1-520494

• **ENGLAND**

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

• **FRANCE**

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

• **GERMANY**

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

• **HOLLAND**

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

• **IRELAND**

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

• **ITALY**

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

• **JAPAN**

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

• **NORWAY**

Bergen Chapter
Kjell Birger Faeraas, 47-518-7784

• **REPUBLIC OF CHINA**

(R.O.C.)
Contact Ching-Tang Tzeng
P.O. Box 28
Lung-Tan, Taiwan 325

• **SWEDEN**

Swedish Chapter
Hans Lindstrom, 46-31-166794

• **SWITZERLAND**

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

SPECIAL GROUPS

• **Apple Corps Forth Users Chapter**

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

• **Baton Rouge Atari Chapter**

Chris Zielewski (504) 292-1910

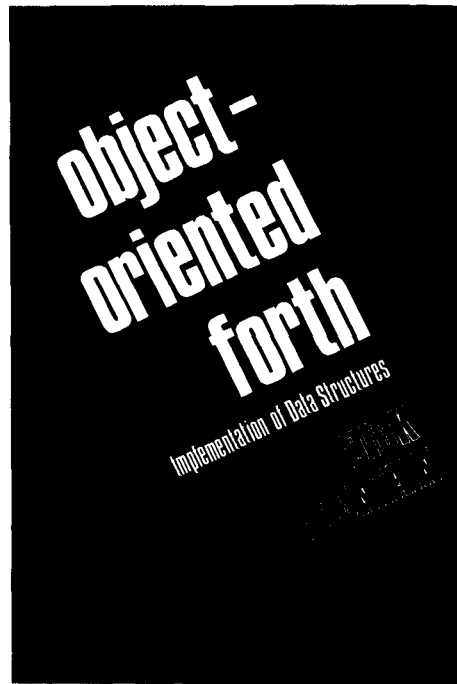
• **FIGGRAPH**

Howard Pearlmutter
(408) 425-8700

• **NC4000 Users Group**

John Carpenter (415) 960-1256

NOW AVAILABLE



\$20 EACH



FROM THE FORTH INTEREST GROUP

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