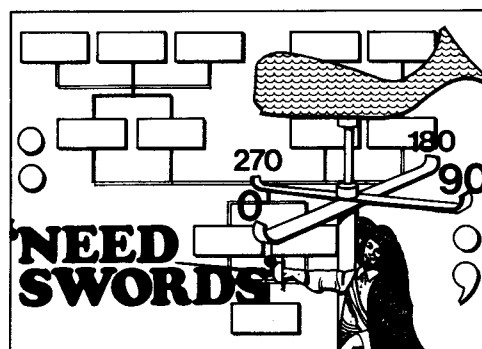


FORTH

Dimensions

Volume 5, Number 3
September/October 1983
\$2.50

8088 RAMdisk



FEATURES

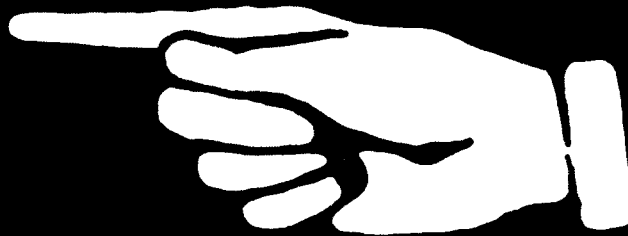
FIG-Forth Vocabulary Structure.....	Evan Rosen.....	5
An Easy Directory System.....	Wil Baden.....	11
A RAMdisk for 8086/8088 FIG-Forth.....	John Irwin.....	14
In-word Parameter Passing.....	Timothy Huang.....	19
Stack-Oriented Co-Processors and Forth.....	Dana Redington.....	20
Code and Colon Compatibility.....	David Held.....	23
CORDIC Algorithm Revisited.....	Dave Freese.....	24
Forth-83: A Minority View.....	Glenn Tenney.....	27

DEPARTMENTS

Letters.....	3	
Editorial: Standard Fare.....	3	
Standards Corner.....	Robert L. Smith.....	26
Techniques Tutorial: Meta Compiling III.....	Henry Laxen.....	31
FIG Chapter News.....	John D. Hall.....	34
New Product Announcements.....	36	

What Do All Have In Common?

Hewlett Packard
AT & T Long Lines
General Electric
Hughes Aircraft
Motorola
Rockwell International
U.S. Army ET & D Labs
U.S. Navy NOSC
University Of California
and over 200 others . . .



Over the past three years, each has bought professional

68000 FORTH

systems from

Creative Solutions Inc.

Why?

MATURE RELIABLE PRODUCT

First Multi-FORTH™ installation in December 1979 — installed base of over 200 sites.

MULTITASKING

Since the beginning, Multi-FORTH™ has supported multiple background tasks and optional multiple users.

16 OR 32 BIT IMPLEMENTATIONS

16 bit 79 — Standard or 32 bit unlimited program size implementations available.

FAST . . .

Eratosthenes Sieve Benchmark in under 18 seconds for 10 passes in high level.

IN-LINE ASSEMBLER

BUILT-IN TRACE, DEBUG FEATURES

CORE IMAGE SNAPSHOT FEATURES

Saves and restores current system image without recompiling (for turnkey applications).

EXTENSIVE DOCUMENTATION

Current user manual is over 350 pages.

ONLINE CAI COURSE, HELP FEATURES

GRAPHICS AND FLOATING POINT, SCREEN EDITORS, ON HP SERIES 200 OR MOTOROLA VME/10

MOST SINGLE BOARD COMPUTERS ARE SUPPORTED

VME110, KDM, ECB, VM01, VM02,
OB68K, BRI, DUAL, ERG, CP/M68K
installations — 8" media.

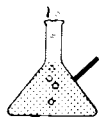
PRICES START at \$1,295 for a SINGLE COMPUTER LICENSE (HP Series 200 32 bit version)

OK!!! I'm interested! Please send me more information about the Multi-FORTH™ system.

Name _____ Company _____

Address _____

Phone _____ Hardware Type _____



Creative Solutions Inc.

4801 Randolph Road
Rockville, Maryland 20852
(301) 984-0262

Multi-FORTH™ is a registered trademark of Creative Solutions, Inc.

FORTH Dimensions

Published by FORTH Interest Group

Volume V, No. 3

September/October 1983

Editor

Marlin Ouverson

Publisher

Roy C. Martens

Typesetting/Production

LARC Computing, Inc.

Cover Art

Al McCahon

FORTH Dimensions solicits editorial material, comments and letters. No responsibility is assumed for accuracy of material submitted. Unless noted otherwise, material published by the FORTH Interest Group is in the public domain. Such material may be reproduced with credit given to the author and the FORTH Interest Group.

Subscription to FORTH Dimensions is free with membership in the FORTH Interest Group at \$15.00 per year (\$27.00 foreign air). For membership, change of address and/or to submit material, the address is: FORTH Interest Group, P.O. Box 1105, San Carlos, CA 94070.

Letters to the Editor

Unfinished Business

Dear Editor,

I am a novice to Forth, but a programmer and analyst for small companies, especially in engineering and scientific work. I would like to see more and better information and programs for new Forth programmers.

Since Forth textbooks are scarce, I learn mostly from *Forth Dimensions*. Many programs are not easy because often they are not fully compatible with my software, or contain words that are not defined. In one program (from "A Techniques Tutorial on Defining Words," Vol. IV, No. 1), there is only one word, **WITHIN**, that I am not able to define at all. I have checked three different textbooks, but the program is still left unfinished.

I would like *Forth Dimensions* to evaluate Forth software like other magazines (e.g., *InfoWorld*). I had to buy three different Forths before finding one I feel comfortable with. I need more information on each piece of software, like which follows the Forth Standard, disk format supported, and options like eighty-column screen, lower-case, editor, etc. These would save my time and money.

Thank you for your time and patience with this comment about publication and evaluation.

Yours sincerely,

William A. Paine
11025 - 131st Ave. NE
Kirkland, WA 98033

(Continued)

Editorial

Standard Fare

Once a month, a pioneering group of Forth aficionados meets to coordinate the considerable business of running a world-wide organization. That it is a not-for-profit affair does not make their duties less complex than those associated with any international business. That the board members are unpaid does not make them less committed, diligent, and effective as managing leaders.

The work of Forth Interest Group members has been largely responsible for the growing public acceptance of Forth. They have called attention to Forth as a practical language and, for more and more projects, as the language of preference. Whenever elements of the language have posed obstacles, they have contributed hours of labor to modify, argue, test, debate, and re-modify to create an improved Forth standard.

Forth-83 has been accepted as the official standard. Two articles in this issue provide a summary of some of the changes that have been introduced, and of some objections that have been raised. Our purpose in publishing these items is to show some of the changes that have been introduced and to let readers see at least part of the process (as well as the importance) of arriving at a new standard.

Of course, the people responsible for all this are just FIG members who get involved. There is always room for another contributor to this loose-knit band. Particularly welcome are articles, ideas, and letters to the editor from the many new members receiving *Forth Dimensions* this year. Let us know how we can help you, and let others know how Forth can help them!

Meanwhile, make good use of this issue and the ones to come. Articles and

code are still being accepted for our issues on data acquisition, instrument control, and math. Utilities and useful applications are always welcome. Writers guidelines are available to authors (and potential authors) who send a self-addressed, stamped envelope to:

Editor
Forth Dimensions
Forth Interest Group
P.O. Box 1105
San Carlos, CA 94070

We look forward to hearing from each of you!

—Marlin Ouverson
Editor

FORTH

for VICTOR 9000
Microcomputer

DAI-E Chinese Language
Computing System Including:

- 5000 most common Chinese characters-can be transmitted in accordance with established CCC11 Communications Code.
- Chinese word processor
- Chinese Forth

Available Fourth Quarter 1983-Call For Price

Dai-E FORTH Level I

Beginner's Package in
Fig-FORTH Style

US \$150⁰⁰

Dai-E FORTH Level II

Professional Level FORTH
Package

Conforms with proposed 1983 standard

Features:

On line Documentation,
Decompiler, Debugger (tracer),
Viewer (help), Line Editor
and Screen Editor, 8086/8088
Assembler, Meta Compiler,
Double precision Math
extensions, Native Operating
System file handler, True LRU
disk buffer mechanism, Separate
header, Graphics/Sound
Interface, Hashed dictionary
structure, Multi-tasking

Available for CP/M, MS-DOS, or
stand-alone versions.

US \$350⁰⁰

Coming Soon

DAI-E GRAPHICS with optical mouse
Available fourth quarter 1983

SEE US AT BOOTH #16

FORTH INTERNATIONAL CONFERENCE
Oct. 14 & 15, 1983 — PALO ALTO, CA.



DAI-E
SYSTEMS
INC.

MULTI-LANGUAGE
COMPUTING SYSTEMS

503/682-3201

29783 Town Center Loop West • P.O. Box 790
Wilsonville, Oregon 97070 U.S.A.

RPN Blues — Revisited

Dear FIG:

I have been trying to implement Forth on my system for two years now, but failed because of not having a good assembler for my system. In those two years I mostly did not work on the Forth system because of frustration. But now I had the opportunity to work with Forth on a friend's machine. Super!

There is one thing I think could provide an improvement in readability of Forth programs: do the control structures have to be in reversed polish notation, or wouldn't it fit in the Forth concept otherwise? How about control structures as below:

FIG-Forth 79

DO (+)LOOP

(cond) **IF** (true) **THEN**
(cond) **IF** (true) **ELSE** (false) **THEN**

BEGIN (cond) UNTIL

BEGIN (cond) WHILE (true) REPEAT

Other version

kept as it is

IF (cond) **THEN** (true) **ENDIF**
[**IF** is only documentary, **THEN** checks
condition
ELSE is as before]

BEGIN UNTIL (cond) **FULFILLED**
[**BEGIN** is where to jump; **UNTIL** is
documentary; **FULFILLED**
is formerly **UNTIL**]

WHILE (cond) **REPEAT** (true) **ENDWHILE**
[**WHILE** marks where to jump; **REPEAT**
checks if cond is true;
ENDWHILE jumps to **WHILE**

What do you think about it?

Horst G. Kroker
HCH-V-Meissen Str. 37
Mainz LL2 6500
W. Germany

Model Behavior

Dear FIG:

While working with a FIG-Forth system, I found a couple of things which may be of interest for inclusion in other compilers. First, there is a bug in the model's implementation of the logic associated with **?PAIRS** which allows the construct

```
IF . . . ELSE . . . ELSE . . .  
ELSE . . . THEN
```

to be compiled without error. The execution of the resulting code is entertaining, but not particularly useful. I would suggest fixing it via the following changes to the model:

Screen 40:

```
: ?PAIRS AND 0= 13 ?ERROR ;
```

Screen 73:

```
: ENDIF ?COMP 6 ?PAIRS HERE OVER - SWAP ! ; IMMEDIATE  
: DO COMP 1 (DO) HERE 8 ; IMMEDIATE  
: LOOP 8 ?PAIRS COMP 1 (LOOP) BACK ; IMMEDIATE  
: +LOOP 8 ?PAIRS COMP 1 (+LOOP) BACK ; IMMEDIATE
```

Screen 74:

```
: ELSE 2 ?PAIRS COMP 1 BRANCH HERE 0,  
SWAP 2 [COMP 1] ENDIF 4 ; IMMEDIATE
```

Use of a bit-masked test thus allows **THEN** to follow either **IF** or **ELSE** but only allows **ELSE** to follow **IF**, which is what we want.

The other thing is a compiler speed-up enhancement. I had always wondered why the dictionary search scanned each entry character by character, even though the length was known, but just chalked it up to one of the mysteries of Forth that I'd figure out some day. It should be noted, incidentally, that I always use a **WIDTH** of thirty-one.

Just recently, however, I found out how things work with a width of less than thirty-one (I think) so I see the basic reason. However, I believe that things could be speeded up dramatically by just using the lower of **WIDTH** and the length found in the dictionary header as an increment to skip to the end of the name. I use a 6801-based FIG system which was so modified and the compile times went down by over thirty percent for one system (around 300 screens, would you believe).

Best regards,

Mike Armstrong
7502 S.W. 143rd Ave.
Miami, FL 33183

(Continued on page 29)

FIG-Forth Vocabulary Structure

*Evan Rosen
Bayside, New York*

Vocabulary structure and linking in FIG-Forth is a clever and complex affair. FIG-Forth makes extensive use of the linked list in vocabulary management and creates a structure that allows the dumb primitive (**FIND**) to look in the right places without (**FIND**) ever realizing it. This note attempts to make both the creation and search processes a little clearer.

Before explaining how vocabulary structures work, let's talk about what they do.

You may recall that during typical dictionary searches, first the context and then the current vocabularies are searched. These two searches are performed in the same way. Take the context search as an example, first looking at the broad picture and then the details.

Vocabulary Search (big picture)

Assume we have the vocabulary "tree" shown in Figure One, and that the vocabulary **NEWVOC** has had a few words added to it. Assume that **NEWVOC** is the context vocabulary. When context is searched, first **NEWVOC** is searched, then **VOC2**, then **VOC1**, and then **FORTH**, assuming no match has been found. **VOC3** is not searched. Thus, the context vocabulary is actually composed of a sequence of vocabularies. The word "vocabulary" itself is, therefore, somewhat ambiguous in FIG-Forth usage.

After a few setup details, the actual search is done by the not-very-smart primitive (**FIND**), which returns only on a match, or on finding a zero for the next name field address in the search. (The zero shows up in the name field of the first word in **FORTH**, usually **LIT**. Try ' **LIT LFA ?**). Hence (**FIND**) has somehow to be guided in order to search all the right vocabularies. This is where Dummy Name Fields, containing the two bytes 81 and A0, come in. To understand the details we have to look at the structure of a vocabulary word.

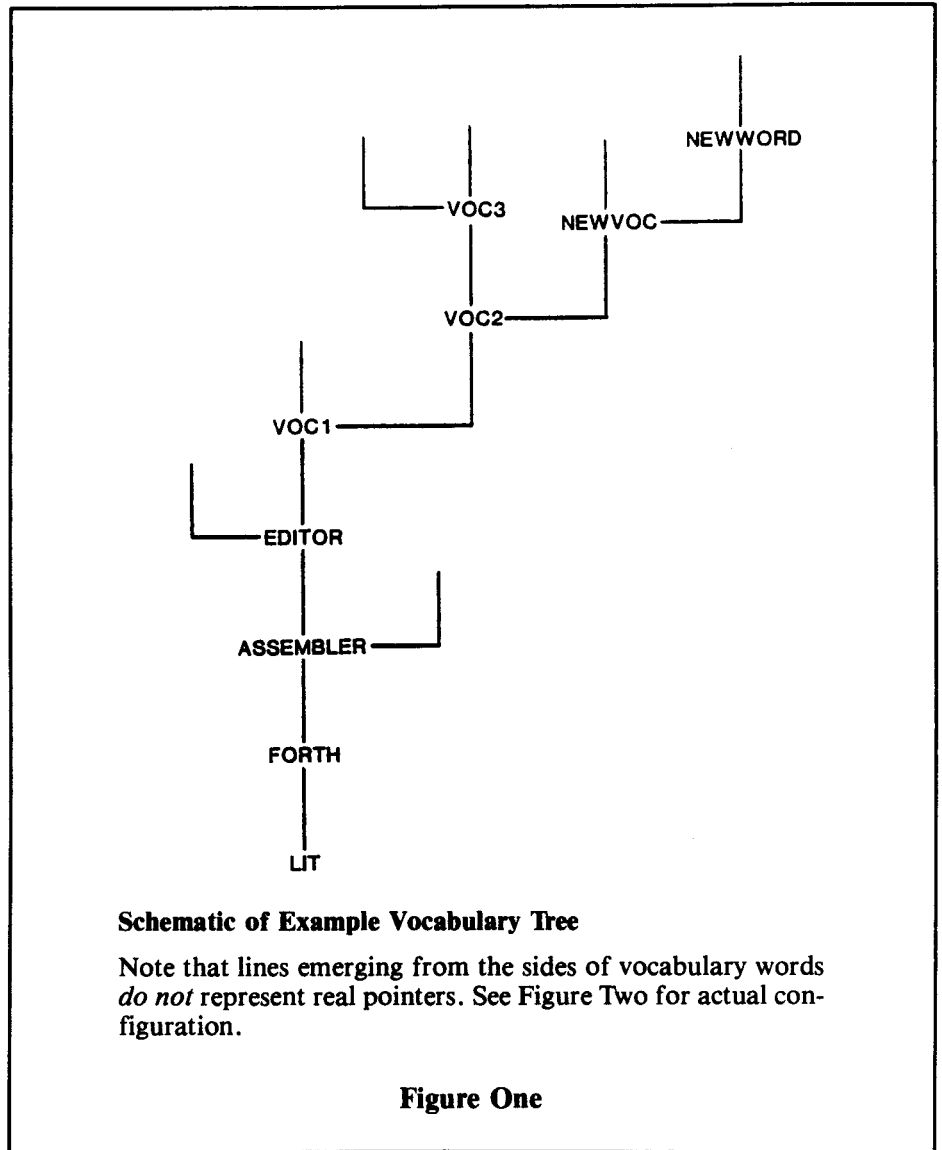
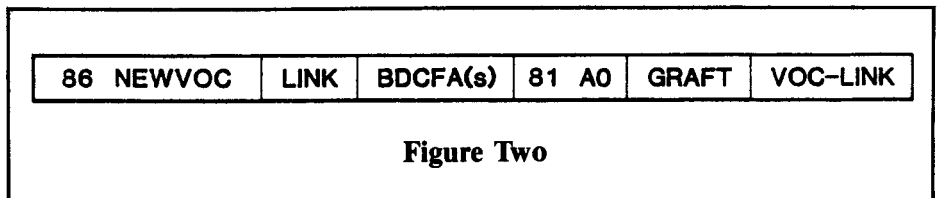


Figure One

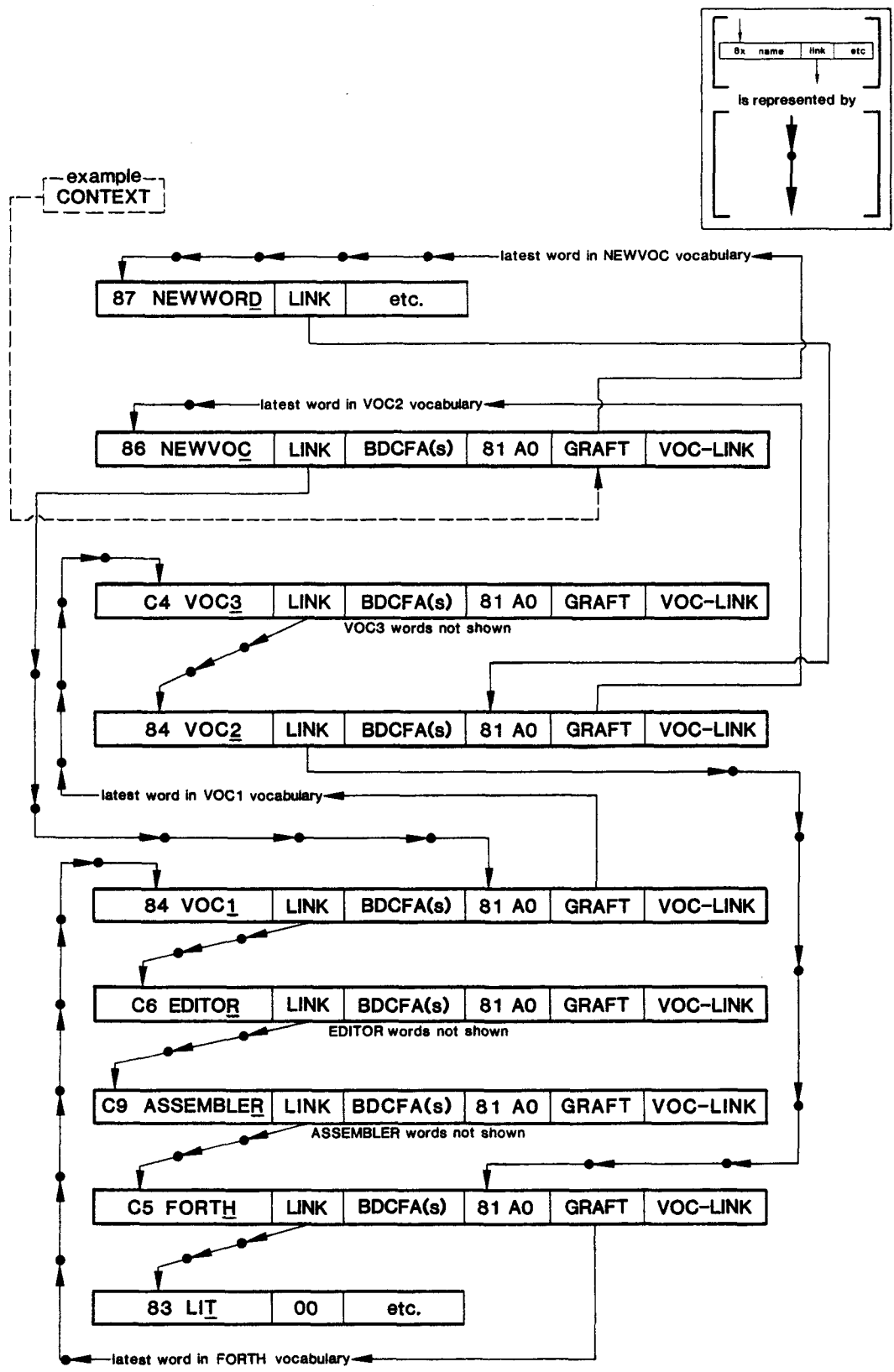


Vocabulary Searches (detailed picture)

Performing **VOCABULARY NEWVOC** will create the elements shown in Figure Two in the dictionary. Taking these items in order, we have:

86

The length byte of the new word, with high bit set, for detection by **TRAVERSE**. If **NEWVOC** were immediate, bit 6 would be set also, making this byte C6.



Detailed Structure of Example Vocabulary Tree

Figure Three

NEWVOC

ASCII of the new word's name. High bit of last character will be set, with very occasional machine-dependent variations, *e.g.*, on 6502 systems.

LINK

Link back to name field address of the previous word defined in the current vocabulary, *i.e.*, the vocabulary in which the word **NEWVOC** is defined. All normal so far.

BDCFA(s)

The **<BUILDS...DOES>** code field address(es). This field is generally four bytes long, though the shorter, faster two-byte renderings are gaining prominence. This need not concern us now. You can tell how long this field is by looking for 81 A0 which will follow it.

81 A0

This is how A081H, will show up in the dictionary. It is a Dummy Name Field with name of length one (the 1 in 81) and actual ASCII name 20 with high bit set, to become A0. (ASCII 20H is a blank, which was chosen because it was rather unlikely to occur as an actual name in a working system. Back to this in a moment.)

GRAFT

It's not clear if this field has another name, but calling it the Graft Field is useful for the moment, as this field helps in "grafting" the new vocabulary onto the vocabulary tree. The Graft Field in a vocabulary-name word like **NEWVOC** or **FORTH** is the actual field that is pointed to when we say something like, "CONTEXT points to **NEWVOC**."

Right after **NEWVOC** is defined, the Graft Field of **NEWVOC** points to the Dummy Header Field of the vocabulary in which it was defined. In the tree

in Figure One, for example, **NEWVOC** was defined in the vocabulary **VOC2**. This is caused by the action of the **<BUILDS** part of the word **VOCABULARY**, when it performs

CURRENT @ CFA,

(This is how the standard is written. **CFA** is misleadingly used as slang for 2 - and should be replaced.)

When the first word, call it **NEWWORD**, in the vocabulary **NEWVOC** is defined, its LF (Link Field) takes the value in the Graft Field, and the Graft Field takes a new value, namely, the NFA (Name Field Address) of **NEWWORD**. The trick is that this is accomplished in the usual way by **CREATE**, which looks at where **CURRENT** is pointing when **CREATE** is ready to set up the new links for **NEWWORD**. **CREATE** then gets the value in the Graft Field (which in our example points to the Dummy Name Field in **VOC2**), and puts this into the link field of **NEWWORD**. The graft has then been created. More in a moment.

VOC-LINK

This points to the voc-link field of the previously defined vocabulary. For the bottom vocabulary, generally **FORTH**, this (**VOC-LINK**) will be 0 to indicate the end of the list. We're not going to talk about voc-links here.

Okay, now, let's see what happens when (**FIND**) unsuccessfully searches the context vocabulary, **NEWVOC**. Assume that some setup routine has properly arranged both the stack and the string that (**FIND**) will be trying to find. The address where (**FIND**) will start looking will be at the top of the stack. In this case it will be **CONTEXT @ @**, since **CONTEXT** points to the Graft Field of **NEWVOC** which points to the Name Field of the last word defined in **NEWVOC**. Then (**FIND**) starts looking.

When (**FIND**) reaches the first word defined in **NEWVOC**, which you recall was **NEWWORD**, (**FIND**) again fails to find a match and so looks in the Link Field of **NEWWORD** to find out where to search next. What is there, if you recall, is the address of the Dummy Name Field of **VOC2**. The unsuspecting (**FIND**) then looks at this field, where it sees the "name" 81 A0, again fails to match, and so goes to what it thinks is the link field corresponding to this

C64-FORTH

for the
Commodore 64

FORTH SOFTWARE FOR THE COMMODORE 64

C64-FORTH (TM) for the Commodore 64 - \$99.95

- Fig Forth-79 implementation with extensions
- Full feature screen editor and macro assembler
- Trace feature for easy debugging
- 320x200, 2 color bit mapped graphics
- 16 color sprite and character graphics
- Compatible with VIC peripherals including disks, data set, modem, printer and cartridges
- Extensive 144 page manual with examples and application screens
- "SAVETURNKEY" normally allows application program distribution without licensing or royalties

C64-XTEND (TM) FORTH Extension for C64-FORTH - \$59.95

- (Requires original C64-FORTH copy)
 - Fully compatible floating point package including arithmetic, relational, logical and transcendental functions
 - Floating point range of 1E+38 to 2E-39
 - String extensions including LEFT\$, RIGHT\$, and MID\$
 - BCD functions for 10 digit numbers including multiply, divide, and percentage. BCD numbers may be used for DOLLAR.CENTS calculations without the round-off error inherent in BASIC real numbers.
 - Special words are provided for inputting and outputting DOLLAR.CENTS values
 - Detailed manual with examples and applications screens
- (Commodore 64 is a trademark of Commodore)

TO ORDER - Specify disk or cassette version

- Check, money order, bank card, COD's add \$1.50
- Add \$4.00 postage and handling in USA and Canada
- Mass. orders add 5% sales tax
- Foreign orders add 20% shipping and handling
- Dealer inquiries welcome

PERFORMANCE MICRO PRODUCTS

770 Dedham Street, S-2
Canton, MA 02021
(617) 828-1209



Next-Generation
Micro-Computer Products

THE FORTH SOURCE™

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. This public domain product includes an editor, FORTH assembler, tools, utilities and the vocabulary for the best selling book "Starting FORTH". The Programmer's Kit provides a complete FORTH for a number of computers. Other MVP-FORTH products will simplify the development of your applications.

MVP Books - A Series

- Volume 1, All about FORTH** by Haydon. MVP-FORTH glossary with cross references to fig-FORTH. *Starting FORTH* and FORTH-79 Standard. 2nd Ed. \$25
- Volume 2, MVP-FORTH Assembly Source Code.** Includes CP/M®, IBM-PC®, and APPLE® listing for kernel \$20

MVP-FORTH Software - A Transportable FORTH

- MVP-FORTH Programmer's Kit** including disk, documentation, Volumes 1 & 2 of MVP-FORTH Series (*All About FORTH*, 2nd Ed. & *Assembly Source Code*), and *Starting FORTH*. Specify CP/M, CP/M 86, CP/M+, APPLE, IBM PC, MS-DOS, Osborne, Kaypro, H89/Z89, Z100, TI-PC, MicroDecisions, Northstar, Compupro, Cromemco \$150
- MVP-FORTH Cross Compiler** for CP/M Programmer's Kit. Can also generate headerless code for ROM or target CPU \$300

- MVP-FORTH Meta Compiler** for CP/M Programmer's kit. Use for applications on CP/M based computer. Includes public domain source \$150
- MVP-FORTH Fast Floating Point** for APPLE Programmer's Kit. Includes 9511 math chip on board with disk and documentation. \$400
- MVP-FORTH Programming Aids** for CP/M, IBM or APPLE Programmer's Kit. Extremely useful tool for decompiling, callfinding, and translating. \$150
- MVP-FORTH** by ECS Software for IBM-PC or ATARI® 400/800. Standalone with screen editor. License required. Upgradeable \$100
- MVP-FORTH** by ECS Software for IBM-PC or ATARI 400/800. Enhanced with color animation, multitasking sound, utilities, and unlimited run time license. \$175
- MVP-FORTH Professional Application Development System (PADS)** for CP/M, IBM-PC, or APPLE. A three level integrated system with complete documentation. Complete system \$400

- MVP-FORTH PADS** enhanced virtual system \$150
- MVP-FORTH PADS** Programming Aids \$150
- MVP-FORTH PADS** Meta Compiler \$150

★★★ MVP-FORTH operates under a variety of CPU's, computers, and operating systems. CP/M® disks can be supplied 8", SS/SD, 3740 format or 5 1/4 for Osborne® Northstar® Micro Decisions® Kaypro® or H89/Z89®. Specify your computer and operating system. ●●●

FORTH DISKS

FORTH with editor, assembler, and manual.

- APPLE** by MM \$100
- APPLE** by Kuntze \$90
- ATARI®** valFORTH \$60
- CP/M®** by MM \$100
- HP-85** by Lange \$90
- HP-75** by Cassidy \$150
- IBM-PC®** by LM \$100
- NOVA** by CCI 8" DS/DD \$150
- Z80** by LM \$50
- 8086/88** by LM \$100

Cartridges by HES \$50
VIC20 \$50 Comm 64 \$60

Enhanced FORTH with: F-Floating Point, G-Graphics, T-Tutorial, S-Stand Alone, M-Math Chip Support, MT-Multi-Tasking, X-Other Extras, 79-FORTH-79.

- APPLE** by MM, F, G, & 79 \$140
- ATARI** by PNS, F, G, & X \$90
- CP/M** by MM, F & 79 \$140
- Apple, GraFORTH** by I \$75
- Multi-Tasking FORTH** by SL, CP/M, X & 79 \$395
- TRS-80/II or III** by MMS, F, X, & 79 \$130
- Timex** by FD, tape G, X, & 79 \$45
- TUTORIAL** by LH, includes *Starting FORTH* \$95
- Extensions** for LM Specify IBM, Z80, or 8086
 - Software Floating Point \$100
 - 8087 Support (IBM-PC or 8086) \$100
 - 9511 Support (Z80 or 8086) \$100
 - Color Graphics (IBM-PC) \$100
 - Data Base Management \$200
- Victor 9000** by DE, G, X \$150

fig-FORTH Programming Aids for decompiling, callfinding, and translating. CP/M, IBM-PC, Z80, or Apple \$150

CROSS COMPILERS Allow extending, modifying and compiling for speed and memory savings, can also produce ROMable code. *Requires FORTH disk.

- CP/M \$300
- 8086 \$300
- Northstar \$300
- IBM \$300
- Z80 \$300
- Apple II/II+ \$300
- FORTH Computer - Jupiter Ace** \$150
 - 16K RAM Pack \$50
 - 48K RAM Pack \$125
 - Par/Sec Interface \$100

Key to vendors:

- CCI Capstone Computing Inc.
- DE Dai-E Systems
- FD Forth Dimension
- I Insoft
- LH Laxen and Harris
- LM Laboratory Microsystems
- MM MicroMotion
- MMS Miller Microcomputer Services
- NS Nautilus Systems
- PNS Pink Noise Studio
- SL Shaw Labs

FORTH MANUALS, GUIDES & DOCUMENTS

- ALL ABOUT FORTH** by Haydon. See above. \$25
- FORTH Encyclopedia** by Derick & Baker. Programmer's manual to fig-FORTH with FORTH-79 references. Flow charted, 2nd Ed. \$25
- Understanding FORTH** by Reymann \$3
- FORTH Fundamentals, Vol. I** by McCabe \$16
- FORTH Fundamentals, Vol. II** by McCabe \$13
- Beginning FORTH** by Chirlian \$17
- FORTH Encyclopedia Pocket Guide** \$7
- And So FORTH** by Huang, A college level text. \$25
- FORTH Programming** by Scanlon \$17
- FORTH on the ATARI** by E. Floegel \$8
- Starting FORTH** by Brodie. Best instructional manual available. (soft cover) \$18 (hard cover) \$23
- 1980 FORML Proc.** \$25
- 1981 FORML Proc 2 Vol.** \$40
- 1982 FORML Proc.** \$25
- 1981 Rochester FORTH Proc.** \$25
- 1982 Rochester FORTH Proc.** \$25
- 1983 Rochester FORTH Proc.** \$25
- A FORTH Primer** \$25
- Threaded Interpretive Languages** \$23
- META FORTH** by Cassidy \$30
- Systems Guide to fig-FORTH** \$25
- Invitation to FORTH** \$20
- PDP-11 User Man.** \$20
- FORTH-83 Standard** \$15
- FORTH-79 Standard** \$15
- FORTH-79 Standard Conversion** \$10
- NOVA fig-FORTH** by CCI Source Listing \$15
- NOVA** by CCI User's Manual includes editor, assembler, and utilities \$25
- Installation Manual for fig-FORTH** \$15

Source Listings of fig-FORTH, for specific CPU's and computers. The Installation Manual is required for implementation. Each \$15

- 1802 6502 6800 AlphaMicro
- 8080 8086/88 9900 APPLE II
- PACE 6809 NOVA PDP-11/LSI-11
- 68000 Eclipse VAX Z80

Ordering Information: Check, Money Order (payable to MOUNTAIN VIEW PRESS, INC.), VISA, MasterCard. COD's \$5 extra. No billing or unpaid PO's. California residents add sales tax. Shipping costs in US included in price. Foreign orders, pay in US funds on US bank, include for handling and shipping by Air: \$5 for each item under \$25, \$10 for each item between \$25 and \$99 and \$20 for each item over \$100. Minimum order \$15. All prices and products subject to change or withdrawal without notice. Single system and/or single user license agreement required on some products.

DEALER & AUTHOR INQUIRIES INVITED

MOUNTAIN VIEW PRESS, INC.

PO BOX 4656

MOUNTAIN VIEW, CA 94040

(415) 961-4103

Dummy Name Field. What it finds instead is the Graft Field of **VOC2**. (**FIND**) then looks in this field, gets the pointer to the Name Field of the last-defined work in **VOC2**, and continues its search. This same tricking of (**FIND**) occurs at each intersection in the tree until (**FIND**) finally ends up at the base of the tree, in the **FORTH** vocabulary, at the Link Field of **LIT**, where it finds a zero and exits.

The term "vocabulary" has been carried over from pre-FIG Forths, where it had a meaning closer to what one would expect. A more descriptive name for the FIG-Forth version might well be **VOCABULARY-BRANCH**.

To review, in the current setup in FIG-Forth,

(1) Dictionary searches may repeatedly search various vocabularies within one search. For instance, the **FORTH** vocabulary is generally searched twice.

(2) Dictionary searches search all of each vocabulary-branch through which they pass, not just the part "below" the intersection. "Chronology" of definitions does not, *per se*, determine the search path.

Where does this lead us? The structure can be customized, to an extent, once it is understood: for instance, storing a zero into a word's link field can stop a search, or redirecting a link can alter the search pattern. Remember, though, that some definition for the word whose name is the null character must remain in the search chain, or the system won't know how to deal with the end of a line. The usual definition is next to that of **QUERY**, or,

```
HEX 8081 HERE
: X R> DROP ; ! IMMEDIATE
DECIMAL
```

compiled above where you're going to zero a link field should allow you to experiment from the terminal (but not from screens).

There are at least two major shortcomings to the present vocabulary organization:

(1) No pointer is kept to the first word in a new vocabulary, only to the last; hence, rearranging the branches on the vocabulary tree is cumbersome.

(2) The search routine only looks at the "current" and "context" vocabularies, and is thus limited in regard to generalized search patterns.

In my next article, we'll look at some of the proposals for vocabulary structuring.

Illustrative figures were kindly provided by Valpar International. —E.R.

**FOR TRS-80 MODELS 1, 3 & 4
IBM PC, XT, AND COMPAQ**

The MMSFORTH System. Compare.

- The speed, compactness and extensibility of the MMSFORTH total software environment, optimized for the popular IBM PC and TRS-80 Models 1, 3 and 4.
- An integrated system of sophisticated application programs: word processing, database management, communications, general ledger and more, all with powerful capabilities, surprising speed and ease of use
- With source code, for custom modifications by you or MMS.
- The famous MMS support, including detailed manuals and examples, telephone tips, additional programs and inexpensive program updates, User Groups worldwide, the MMSFORTH Newsletter, Forth-related books, workshops and professional consulting.

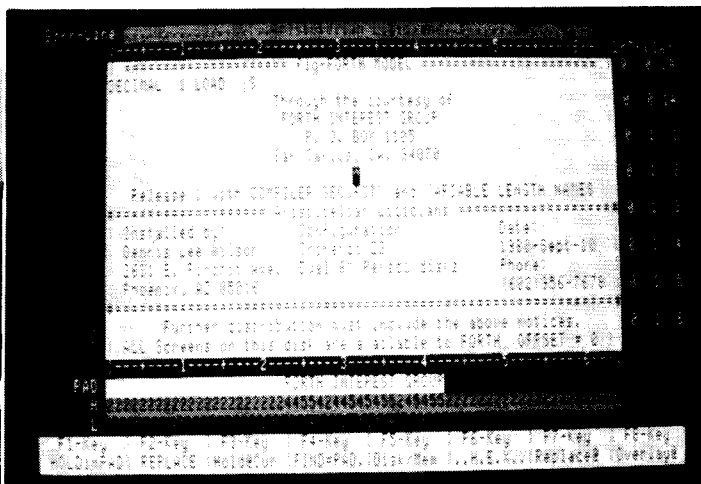
mmsFORTH

A World of Difference!

- Personal licensing for TRS-80: \$129.95 for MMSFORTH, or "3+4TH" User System with FORTHWRITE, DATAHANDLER and FORTHCOM for \$399.95.
- Personal licensing for IBM PC: \$249.95 for MMSFORTH, or enhanced "3+4TH" User System with FORTHWRITE, DATAHANDLER-PLUS and FORTHCOM for \$549.95.
- Corporate Site License Extensions from \$1,000.

If you recognize the difference and want to profit from it, ask us or your dealer about the world of MMSFORTH.

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



8080/Z80 FIG-FORTH for CP/M & CDOS systems FULL-SCREEN EDITOR for DISK & MEMORY

\$50 saves you keying the FIG FORTH model and many published FIG FORTH screens onto diskette and debugging them. You receive TWO diskettes (see below for formats available). The first disk is readable by Digital Research CP/M or Cromemco CDOS and contains 8080 source I keyed from the published listings of the FORTH INTEREST GROUP (FIG) plus a translated, enhanced version in ZILOG Z80 mnemonics. This disk also contains executable FORTH.COM files for Z80 & 8080 processors and a special one for Cromemco 3102 terminals.

The 2nd disk contains FORTH readable screens including an extensive FULL-SCREEN EDITOR FOR DISK & MEMORY. This editor is a powerful FORTH software development tool featuring detailed terminal profile descriptions with full cursor function, full and partial LINE-HOLD LINE-REPLACE and LINE-OVERLAY functions plus line insert/delete, character insert/delete, HEX character display/update and drive-track-sector display. The EDITOR may also be used to VIEW AND MODIFY MEMORY (a feature not available on any other full screen editor we know of.) This disk also has formatted memory and I/O port dump words and many items published in FORTH DIMENSIONS, including a FORTH TRACE utility, a model data base handler, an 8080 ASSEMBLER and a recursive decompiler.

The disks are packaged in a ring binder along with a complete listing of the FULL-SCREEN EDITOR and a copy of the FIG-FORTH INSTALLATION MANUAL (the language model of FIG-FORTH, a complete glossary, memory map, installation instructions and the FIG line editor listing and instructions).

This entire work is placed in the public domain in the manner and spirit of the work upon which it is based. Copies may be distributed when proper notices are included.

	USA	Foreign AIR
<input type="checkbox"/> FIG-FORTH & Full Screen EDITOR package		
Minimum system requirements:		
80x24 video screen w/ cursor addressability		
8080 or Z80 or compatible cpu		
CP/M or compatible operating system w/ 32K or more user RAM		
Select disk format below, (soft sectored only)	\$50	\$65
<input type="checkbox"/> 8" SSSD for CP/M (Single Side, Single Density)		
Cromemco CDOS formats, Single Side, S/D Density		
<input type="checkbox"/> 8" SSSD <input type="checkbox"/> 8" SSDD <input type="checkbox"/> 5 1/4" SSSD <input type="checkbox"/> 5 1/4" SSDD		
Cromemco CDOS formats, Double Side, S/D Density		
<input type="checkbox"/> 8" DSSD <input type="checkbox"/> 8" DSDD <input type="checkbox"/> 5 1/4" DSSD <input type="checkbox"/> 5 1/4" DSDD		
Other formats are being considered, tell us your needs.		
<input type="checkbox"/> Printed Z80 Assembly listing w/ xref (Zilog mnemonics)	\$15	\$18
<input type="checkbox"/> Printed 8080 Assembly listing	\$15	\$18

TOTAL \$ _____

Price includes postage. No purchase orders without check. Arizona residents add sales tax. Make check or money order in US Funds on US bank, payable to:

Dennis Wilson c/o
Aristotelian Logicians
2631 East Pinchot Avenue
Phoenix, AZ 85016
(602) 956-7678

An Easy Directory System

Will Baden
Costa Mesa, California

A problem many have with Forth is remembering where things are located. The usual solution is to prepare a directory, *i.e.*, a screen with the names of things and the number of the screen which contains the thing named; or, if you are lucky enough to have a printer, you can list a hard copy of an index which contains the first line of every screen. Then you eyeball it for what you want.

A better way is to let Forth do the lookup for you. Define a word which will search the directory, find the screen number for you, and then push it on the stack. With this value, you can list it or do anything else you want with it.

NAME word LIST

This will list the screen where "word" is located.

A directory consists of alternating names (or subjects) and associated values in free format, beginning on the second line of each screen. The first line is reserved for heading and date. (To begin with line 0 instead of line 1, change C/L to 0 in **CONNIVE**.)

A fragment of a directory is shown in Figure One and will be used in our examples.

```
SCR # 78
( DIRECTORY      WWB/WWB 830714 )
INITIAL 30 STARTING-FORTH 32
KERNEL'S 96 SOLO 98 TRACE 89
FORMATTER 50      PRINT 66
DISCARD 100 RECREATE 100 RELOAD 100
PHONES 49
DOC-DIR 45 DOC-DIR-END 47

SORT 130 SORTED 130 ID< 131 NAMES 132
ALLNAMES 132 .NAMES 132
SWORDS 133
```

Figure One

Of course, the values in a directory do not have to be screen numbers. They could be anything that you want, *e.g.*, phone numbers, part numbers, or operating system constants.

Directories do not have to be in a neat order, and the user is responsible for maintaining them. Obviously, it is

helpful if there is some semblance of order. For best performance, the most frequently used words should be at the beginning.

If you want to list the screen where the word **SWORDS** is located, then all you have to do is type **NAME SWORDS LIST** or you can say **VIEW SWORDS** and the result will be the same. **VIEW** is defined

```
: VIEW ( ---<name> )
  NAME LIST ;
```

This will list screen 133 where **SWORDS** is defined. The word **SWORDS** after **NAME** or **VIEW** is sought in the directory, the number 133 which follows the word **SWORDS** is pushed onto the stack, and screen #133 is listed. You can define other words like **VIEW** to perform any operation with the number on the stack.

If you want to load that screen, then another word named **NEED** can be used, *e.g.*,

NEED SWORDS

This will search the current working directory screen and then load the screen which has **SWORDS** defined on it. It checks to see if the word which is needed has been already defined; if so, it will not be loaded again. This word is very useful, since it can load other screens when and where they are needed. (See Figure Two.)

If any of the words needed are not already defined, they will be loaded before the rest of the screen. This way you can load the screens in any order as long as you have stated *on each screen* what words need to be loaded before that screen.

When screen 133 is loaded it checks for **SORT**, **ID<** and **NAMES**. If any of these is not defined it will load screen 130, 131 and/or 132 as appropriate. It then checks for **.NAMES** which just happens to be on the same screen as **NAMES** and so will always be already defined. "NEED something-else" may have been used on needed screens, and so forth.

This way all words will get their location from the directory. If at any time you move the screen to another location, just change the directory to show the proper screen number. Any screen which depends on the word whose screen location has changed will not be affected.

A directory is specified by two screen numbers: starting screen and ending screen. The system remembers these values for the current working directory. **DIR** will list the first screen of the current working directory.

To change the screen numbers of the current working directory, **ESTABLISH** can be used.

```
<starting-scr#> <ending-scr#>
ESTABLISH
```

```
Screen 133 is

SCR # 133
( "SWORDS" SORTED WORDS   WWB/WWB 820317 )

NEED SORT   NEED ID<   NEED NAMES   NEED .NAMES

: SWORDS ( -- )
  NAMES ( A,N )
  DUP CR . ." NAMES DEFINED " CR
  2DUP SORTED ID<
  .NAMES ;
```

Figure Two

The defining word **DIRECTORY** at compile time takes two values from the stack which at run time will be used as arguments for **ESTABLISH**.

```
: DIRECTORY 2CONSTANT DOES>
  2@ ESTABLISH ;
```

You could define a word **DOC** which, when executed, would establish the current working directory for documentation as follows (assuming it to be on 45 through 47):

45 47 DIRECTORY DOC

You can even get the values from the current working directory

```
NAME DOC-DIR
NAME DOC-DIR-END
DIRECTORY DOC
```

assuming that the current working directory has entries

DOC-DIR 45 DOC-DIR-END 47

DOC will change the current working directory to the documentation directory. Any number of working directories can be thus defined.

Every system will have a standard or default working directory. To get back to it we say **MAIN**. On this disk it is defined

78 82 DIRECTORY MAIN

The actual work of **NAME** is done by **SUBJECT**. **NAME** picks up the limits for the current working directory and executes **SUBJECT**. **SUBJECT** takes the next word in the input stream and puts it in **PAD**. It then does **CONNIVE**, which will look for that word on the screens indicated. Only the names are compared — the values are skipped over to make the search faster. **INTRIGUE** is used by **CONNIVE** to look up the word in **PAD** on a single screen. When and if the word in **PAD** is found, the next word will be interpreted.

SUBJECT can be used to define words similar to **NAME** and **VIEW** for special directories. A "help" system could be defined something like:

```
: HELP ( --- )
  HELP% 2@ SUBJECT LIST ;
```

With shadow screens it could be even easier.

```
: HELP ( --- )
  NAME >SHADOW LIST ;
```

```
SCR# 71
( you may already have some of these. )
-1 CONSTANT TRUE 0 CONSTANT FALSE
: DEFINED ( --<name> a,f ) ( -- not --or-- )
  -FIND ( this is figforth "-FIND" )
  IF 64 ( precedence bit ) AND
  IF 1 ( it's immediate ) ELSE -1 THEN
  SWAP CFA SWAP
  ELSE HERE 0 THEN ;
: HAVE ( --<name> f ) DEFINED SWAP DROP 0= NOT ;
( : word word here ; ) ( homonym )
: COMPARE ( a1,a2,n1 -- negative/zero/positive )
  OVER + SWAP
  DO COUNT I C@ - ?DUP
  IF SWAP 0= LEAVE THEN
  LOOP
  IF 0 THEN ;
```

```
SCR# 72
( easy directory system wwb/wwb B30714 )
: MORE ( -- addr,f ) ( bl word dup c@ --or-- )
  BEGIN BL WORD DUP 1+ C@ BL OR BL -
  IF TRUE EXIT THEN
  BLK @ 1+ B/SCR MOD
  WHILE DROP 1 BLK +! 0 >IN !
  REPEAT FALSE ;
: INTERPRET-A-WORD ( --<word> )
  ( interprets a word )
  DEFINED
  IF EXECUTE
  ELSE NUMBER ( dpl @ 0< if drop then )
  THEN ;
: CONTINUED ( n -- ) ( b/scr * ) BLK ! 0 >IN ! ;
73 LOAD 74 LOAD ( directory system )
78 82 DIRECTORY MAIN MAIN
```

```
SCR# 73
( easy directory system wwb/wwb B30714 )
: INTRIGUE ( -- flag ) ( search the screen )
  BEGIN MORE ( addr,f )
  IF PAD DUP C@ 1+ COMPARE ( 0 for equal )
  IF BL WORD 0= ( 0 ) ELSE TRUE EXIT THEN
  THEN
  UNTIL FALSE ;
: CONNIVE ( scr1,scr2 -- ) ( <name> is in "PAD" )
  BLK @ >IN @ >R >R TRUE ROT ROT 1+ SWAP
  DO I ( b/scr * ) BLK ! C/L ( skip top line ) >IN !
  INTRIGUE IF NOT LEAVE THEN
  LOOP "ABORT" not in directory "
  INTERPRET-A-WORD
  R> R> >IN ! BLK ! ;
: SUBJECT ( scr1,scr2,--<name> ) ( find and execute )
  BL WORD COUNT PAD 2DUP C! 1+ SWAP CMOVE CONNIVE ;
```

```
SCR# 74
( easy directory system wwb/wwb B30714 )
2VARIABLE DIR%
: %DIR ( -- scr1,scr2 ) DIR% 2@ ;
: ESTABLISH ( scr1,scr2 -- ) DIR% 2! ;
: DIRECTORY ( scr1,scr2 -- )
  2CONSTANT DOES> ( -- ) 2@ ESTABLISH ;
: NAME ( --<name> n --or-- d ) %DIR SUBJECT ;
: VIEW ( --<name> ) NAME LIST ;
: NEED ( --<name> )
  >IN @ HAVE IF DROP ELSE >IN ! NAME LOAD THEN ;
: FOLLOW ( --<name> )
  >IN @ HAVE IF DROP ELSE >IN ! NAME CONTINUED THEN ;
: RUN ( --<name> ) >IN @ NEED >IN ! ;
: DIR ( --<name> ) %DIR MIN LIST ;
: SUB ( scr --<name> n --or d ) DUP SUBJECT ;
```

PROCEDAMUS WWB 7/14/83

End Listing

A primitive phone list can be set up:

```
: .PH# ( DN --- )  
<##### ASCII - HOLD  
#S #> TYPE SPACE ;  
  
: REACH ( --- )  
[ NAME PHONES ] LITERAL  
DUP SUBJECT .PH# ;
```

Since the value of a name or subject is interpreted, **SUBJECT** could be used for menus. Assuming that **#MENU** is a screen with one- or two-character codes alternating with associated words to process them, something like the following could be done.

```
: MENU ( --- )  
#MENU LIST PROMPT  
QUERY 0 > IN !  
#MENU DUP SUBJECT ;
```

“**RUN** something” is equivalent to “**NEED** something something”.

“**FOLLOW** name” has a somewhat similar relation to “**NEED** name” that “**-->**” has to “**LOAD**”. It is used at the beginning of a screen to get to an earlier screen that will lead to the current screen. This allows an entire application, spanning several screens, to be loaded from a request for any one of the constituent words.

In the original conception, we thought that directories would be

sparse, with only major entries like traditional directory screens. We soon found that all definitions could be put in the directory (and a utility to do this was developed). This makes **VIEW** act like its homonym, which vectors **CREATE**. An important difference is that our **VIEW** can list the source of words that are not yet defined.

Since June 1982, this system has been installed on Apple, Atari, CP/M and Heath Figforth, Micromotion Forth79, MVP Forth, and the Starting Forth dialect. The following utility words or their equivalent are required:

MORE returns (address,true) if there are more words in the input stream, (address,false) if the input stream is exhausted. Our definition should work for any system, even when **B/SCR** is not 1. On a standard system, the definition may be replaced with

```
: MORE ( --- a,f )  
BL WORD DUP C@ ;
```

(See Suralis and Brodie, “Checksum for Hand-Entered Source Screens,” *Forth Dimensions*, Vol. IV, No. 3, p. 15.)

INTERPRET-A-WORD interprets the next word in the input stream. **NUMBER** is like the Starting Forth word, and returns a number or double number. In FIG-Forth you will have to remove the parenthesis marks from **DPL @ 0 < IF DROP THEN**.

DEFINED returns a compilation address (which can be executed) and a true flag if the next word is defined, or a string address (which can be further massaged) and a false flag if the next word is unknown. It can be replaced with **- NOT** in some systems.

HAVE returns **TRUE** or **FALSE** depending on whether you already have the next word or not. In Forth79 you may replace it with **FIND**.

All that is required of **COMPARE** is that strings at **HERE** and **PAD** can be compared for equality. **INTRIGUE** can be adapted to use the Starting Forth **-TEXT** or the FIG-Forth **-TEXT** instead.

CONTINUED is from the reference word set and goes to a screen with no return. It is used in the definition of **FOLLOW**. A “named **-->**” can be done

NAME word **CONTINUED**

If **B/SCR** is not 1 then remove the parentheses from around **B/SCR *** in **CONTINUED** and **CONNIVE**.

First Screen of “FORMATTER” Directory

```
( FMT K&P/WWB 830714 )  
LINES 230 =? 230 DATE 230 .DATE 230 TODAY 230 PAGELEN 231  
PAGEWIDTH 231 HUGE 231 MAXSTRING 231 CURPAGE 232 NEWPAGE 232  
LINENO 232 PLVAL 232 M1VAL 232 M2VAL 232 M3VAL 232 M4VAL 232  
BOTTOM 232 HEADER 232 FOOTER 232 FILLING 233 RJUST 233  
LSVAL 233 SPVAL 233 INVAL 233 RMVAL 233 TIVAL 233 CEVAL 233  
ULVAL 233 OUTP 234 OUTW 234 OUTWDS 234 DIRECTION 234  
BETWEEN 234 CAP 234 ALLCAP 234 OUTBUF 234 DATE 235 .DATE 235  
TODAY 235 PUTTL 235 PUTHEAD 236 PUTFOOT 236 PUTLINE 237 BR 238  
PUTSPACE 238 PUTPAGE 239 GETPARAM 240 SETVAL 240 SETLS 241  
SETCE 241 SETUL 241 GETTL 242 SETPAGE 242 SETSP 242 SETNE 242  
SETIN 243 SETRM 243 SETTI 243 SETBOTTOM 244 SETPL 244  
SETM1 244 SETM2 244 SETM3 244 SETM4 244 .FI 245 .NF 245  
.BR 245 .JU 245 .RJ 245 .NJ 245 .LS 246 .CE 246 .UL 246  
.HE 246 .FO 246 .BP 247 .SP 247 .NE 247 .IN 247 .RM 247
```

Figure Three

A RAMdisk for 8086/8088 FIG-Forth

John W. Irwin
Austin, Texas

My IBM Personal Computer, with 320K of RAM, has far more memory than is used by most Forth programs. The desire to eliminate wear and tear on my diskette drives and diskettes prompted me to develop a RAMDisk application in Forth to make use of this resource. The program measures the unused RAM space remaining after Forth is loaded and makes the free space into a virtual diskette drive of that maximum capacity. The performance increase is impressive and the absence of the usual diskette commotion is welcome. By copying a set of screens to RAMDisk, program changes may be tried non-destructively and then copied back to the original screens when completely debugged.

Non-Standard Words

My Forth, although similar to the FIG 8086 implementation, is a greatly expanded version with dictionaries for multi-tasking, full-screen, color, and provision for DOS-compatible, named disk files. The RAMDisk program presented here is a subset of my program; the omitted material pertains mainly to presenting the RAMDisk to the user as a DOS file.

This program makes use of several words from past issues of *Forth Dimensions*: the Kitt Peak **GODO**, and the modular programming words **INTERNAL**, **EXTERNAL** and **MODULE**. These definitions are presented for reference. The extended segment load and store words **EC@**, **EC!**, **E@** and **E!** in my system address the segment defined by a user variable **EB** (Extended Base). My input/output words are **IO@**, **IO!**, **IOC@**, and **IOC!**. The meaning is obvious and equivalent words may exist in other 8086/88 Forths or may be coded by the user. The assembler words used in **(MEM)** have obvious functions.

```
0 ( Kitt Peak GODO and Modular Programming Words )
1
2
3 : (GODO) 2* 0 MAX R @ 4 - MIN R> DUP DUP @ + >R
4       + 2+ @ EXECUTE ;
5
6 : GODO    COMPILER (GODO) HERE 0 , 2 ; IMMEDIATE
7
8
9 : INTERNAL    CURRENT @ @ ;      ( start private program section)
10
11 : EXTERNAL   HERE ;             ( end private program section)
12
13 : MODULE     PFA LFA ! ;        ( hide the private words)
14
15

R

SCR # 768
0 ( RamDisk Program )          INTERNAL
1
2 0 CONSTANT RAM#              ( total number of RamDisk buffers )
3 16384 CONSTANT RAMBLK        ( first block number )
4 0 CONSTANT RAMSEG            ( base segment address of RamDisk )
5
6 HEX
7
8 ASSEMBLE CODE (MEM) 12 INT PSHAX PSHCS END-CODE
9
10 : RDmemsize ( --- ) ( stores seg base, RAMSEG, and # bfrs )
11       (MEM) 1000 + DUP ' RAMSEG ! >R 40 * R> -
12       11 / ' RAM# ! ;
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506

```

```

SCR # 771
0 ( RamDisk Program )      HEX
1
2 : RDerr CR ." RamDisk ERROR, improper function call" QUIT ;
3
4 : RDread ( bfr blk --- )  ( copy relative block )
5   RDseg                    ( set buffer segment )
6   0 E@ =                   ( see if data present )
7   IF                       ( read actual buffer )
8     102 2 DO                ( copy RamDisk to buffer )
9       I E@ OVER ! 2+
10      2 +LOOP DROP
11   ELSE                     ( RamDisk buffer not valid )
12     100 BL FILL            ( fill buffer with blanks )
13   ENDIF ;
14                               DECIMAL  -->
15

```

```

SCR # 772
0 ( RamDisk Program )
1
2 : RDR/W ( addr blk f --- ) ( augments FORTH R/W )
3   OVER DUP                 ( see if in RamDisk )
4   RAMBLK 1- > SWAP RAM# RAMBLK + < AND
5   IF                       ( then memory read or write )
6     1+ GODD
7     RDerr RDwrite RDread RDerr
8     THEN
9     R> DROP                 ( drop return link to R/W )
10    ELSE                    ( else real read or write )
11    R> SWAP >R >R           ( replace R> from R/W )
12    ENDIF ;                 ( and return to R/W )
13
14                               -->
15

```

```

SCR # 773
0 ( RamDisk Program )
1
2 : RDinit ( --- )          ( initialize a phantom diskette in RAM )
3   RDmemsize RDclear ' RDR/W CFA ' R/W ! ;
4
5 : RDparms ( --- )        ( display RamDisk parameters )
6   DECIMAL CR CR ." RamDisk " CR CR
7   ." Capacity : " RAM# 4 / . ." K-bytes" CR
8   ." Blocks : " RAMBLK .
9   ." Screens : " RAMBLK 4 / DUP . CR
10  ." " " RAM# RAMBLK + 1- . CR
11  ." " " RAM# 4 / + . CR ;
12
13                               -->
14
15

```

```

SCR # 774
0 ( RamDisk Program )
1
2 EXTERNAL
3
4 : RD ( 0 --- initialize RD,
5   1 --- return initial block #,
6   2 --- return number of blocks,
7   >2 --- type information block )
8   1+ GODD RDparms RDinit RAMBLK RAM# RDparms THEN ;
9
10 MODULE FORTH
11
12 0 RD 3 RD ( initialize RD and show user the size )
13
14
15

```

Program Interface

The RAMDisk program should be loaded before other applications. The only nucleus word affected is **R/W**. In order to make existing words, such as **FLUSH**, work with the RAMDisk, this program replaces the first parameter word of **R/W (R>)** with the execution address of **RDR/W**. **RDR/W** checks the block number to see if the call is to RAMDisk. If not, then the **R>** is replaced at **#R/W** is emulated and execution proceeds with the native **R/W** code. Since the RAMDisk program modifies a nucleus word, it should be "sealed" under **FENCE** to prevent accidental **FORGETTING**.

The assembler word (**MEM**) returns two values from which the available memory is determined. The PC-DOS call 12 **INT** returns the memory size in Kbytes. Pushing the **CS** register returns the beginning address of the Forth code segment. From these values, **GET-MEM-SIZE** determines space available for the RAMDisk so that the compiled application can be moved freely between machines with differing memory size. If Forth is loaded at a fixed location in a fixed memory-size machine, the word (**MEM**) can be replaced by a constant for memory size and a constant for the end of the Forth segment.

My Forth uses 256-byte blocks. Each block is assigned seventeen sixteen-byte 8086/88 "paragraphs". This wastes twelve bytes per block but simplifies buffer addressing. The extra segment pointer (**EB**) is set to the starting address of RAMDisk plus seventeen times the relative block number within the RAMDisk. The block identifier is then at offset zero and the block proper at offset two to 257 relative to the segment register (**EB**). This scheme is easily adapted to other buffer sizes. For systems with 1K blocks, each block is assigned sixty-five paragraphs.

Program Functions

RD is the only RAMDisk word visible to the user. This word accepts a function flag as follows:

End Listing

FORTH for Z-80[®] , 8086, 68000, and IBM[®] PC

FORTH Application Development Systems include interpreter/compiler with virtual memory management and multi-tasking, assembler, full screen editor, decompiler, utilities, and 130+ page manual. Standard random access files used for screen storage, extensions provided for access to all operating system functions.

Z-80 FORTH for CP/M [®] 2.2 or MP/M II	\$ 50.00
8080 FORTH for CP/M 2.2 or MP/M II	\$ 50.00
8086 FORTH for CP/M-86 or MS-DOS	\$100.00
PC/FORTH[™] for PC-DOS, CP/M-86, or CCPM	\$100.00
68000 FORTH for CP/M-68K	\$250.00

83 - Standard version of all application development systems available soon. All registered users will be entitled to software update at nominal cost.

FORTH + Systems are 32 bit implementations that allow creation of programs as large as 1 megabyte. The entire memory address space of the 68000 or 8086/88 is supported directly for programs and data.

PC/FORTH + for PC-DOS or CP/M-86	\$250.00
8086 FORTH + for CP/M-86	\$250.00
68000 FORTH + for CP/M-68K	\$400.00

Extension Packages for FORTH systems

Software floating point (Z-80, 8086, PC only)	\$100.00
Intel 8087 support (8086, PC only)	\$100.00
AMD 9511 support (8086, Z-80 only)	\$100.00
Color graphics with animation support (PC only)	\$100.00
Symbolic interactive debugger (PC only)	\$100.00
Cross reference utility	\$ 25.00
PC/GEN [™] (custom character sets, PC only)	\$ 50.00
PC/TERM communications program for PC and Smartmodem	\$ 60.00
Hierarchical file manager	\$ 50.00
B-tree index manager	\$125.00
B-tree index and file manager	\$200.00

QTF + Screen editor and text formatter by Leo Brodie,
for IBM PC with IBM or Epson printer

	\$100.00
--	----------

Nautilus Cross Compiler allows you to expand or modify the FORTH nucleus, recompile on a host computer for a different target computer, generate headerless and ROMable code. Supports forward referencing. Produces executable image in RAM or disk file. No license fee for applications. Prerequisite: Application Development System for host computer.

Hosts: Z-80 (CP/M 2.2 or MP/M II), 8086/88 (CP/M-86 or MS-DOS), IBM PC (PC-DOS or CP/M-86), 68000 (CP/M-68K)
Targets: 8080, Z-80, 8086/88, 6502, LSI-11, 68000, 1802, Z-8

Cross-Compiler for one host and one target	\$300.00
Each additional target	\$100.00

AUGUSTA[™] ADA subset compiler from Computer Linguistics, for Z-80 computers under CP/M 2.2

	\$ 90.00
--	----------

LEARNING FORTH computer-assisted tutorial by Laxen and Harris for CP/M, includes Brodie's
"Starting FORTH" (8" format only)

	\$ 95.00
--	----------

Z-80 Machine Tests Memory, disk, printer, and console tests with all source code in standard Zilog
mnemonics

	\$ 50.00
--	----------

8080 and Z-80 application development systems require 48 kbytes RAM and 1 disk drive, 8086 and 68000 require 64 kbytes. Prices include shipping by UPS or first class mail within USA and Canada. California residents add appropriate sales tax. Purchase orders accepted at our discretion. Master Charge and Visa accepted.

Disk formats available: Standard CP/M 8" SSSD, Northstar 5 1/4" QD, Micropolis 5 1/4" QD, Sage 5 1/4" DD, Apple 5 1/4", Victor 9000 5 1/4", Kaypro 5 1/4", Osborne 5 1/4" DD, Micromate 5 1/4", IBM PC 5 1/4", Standard MS-DOS 5 1/4" SSDD. Most other formats can be special ordered.

Laboratory Microsystems, Inc.
4147 Beethoven Street
Los Angeles, CA 90066
(213) 306-7412

Z-80 is a registered trademark of Zilog, Inc.
CP/M is a registered trademark of Digital Research, Inc.
IBM is a registered trademark of International Business Machines Corp.

Augusta is a trademark of Computer Linguistics
dBASE II is a trademark of Ashton-Tate
PC/FORTH and PC/GEN are trademarks of Laboratory Microsystems Inc.

- 0 **RD** (0 ---)
clears RAMDisk, sets RAMDisk length.
- 1 **RD** (1 --- n)
returns initial block of RAMDisk.
- 2 **RD** (2 --- n)
returns number of blocks in RAMDisk.
- >2 **RD** (3 ---)
types a summary block on the screen.

This function call interface is implemented by **GODO** and is very easy to interface to existing programs.

Clearing the RAMDisk (0 **RD**) should be done before use and is done simply by writing zero to the block numbers in the RAMDisk. This operation checks the memory size and resets the RAMDisk origin and length, allowing the program to dynamically adapt to a different memory size and program load address each time it is used.

RDR/W is equivalent to FIG-Forth **R/W**. The stack at entry contains the standard **R/W** call (addr blk# f). If the block number is in the RAMDisk, a **GODO** is used to interpret the flag, else control is passed back to **R/W** with the flag moved to the R-stack.

RDwrite copies the Forth buffer into the RAMDisk buffer, including the block number. The update flag is never set in RAMDisk buffers since the RAMDisk represents the physical diskette. To provide some (very needed) feedback to the user that his **EDIT** is indeed being saved, a speaker click is emitted for each RAMDisk buffer write. This is done very simply in the two lines containing I/O words by gating and immediately de-gating a bit in the speaker port. These lines can be deleted for other systems or if feedback is not desired.

RDread compares the RAMDisk buffer block number to the requested block number to see if the buffer has been written. A valid buffer is indicated by a match while an invalid buffer contains the zero put there by initialization. If the buffer is valid, the RAMDisk buffer is copied to the Forth buffer, else the Forth buffer is blanked using **FILL**.

5th FORML Conference

November 23-25, 1983
Asilomar Conference Center
Pacific Grove, California, U.S.A.

FORML is a technically advanced conference of FORTH practitioners. The topics to be discussed will affect the future evolution of FORTH. FORTH programmers, managers, vendors, and users will benefit from several informative conference sessions. All attendees are asked to participate and are encouraged to write a paper for presentation in an oral or poster session.

Topics Suggested for Presentation

Hardware FORTH Implementation	Nucleus Variations
Large Address Space Environments	Operating System Environments
Multiprogramming Architectures	System Generation Techniques

Registration and Papers

Complete the registration form, selecting accommodations desired and send with your payment to FORML. Include a 100 word abstract of your proposed paper. Upon acceptance by FORML, a complete author's packet will be sent. Completed papers are due September 30, 1983.

Registration Form

Complete and return with check made out to:
FORML P.O. Box 51351, Palo Alto, Calif. 94303

Name _____

Company _____

Address _____

City _____ State _____ ZIP _____

Phone (day) _____ (evening) _____

I have been programming in FORTH for: (years) _____ (months) _____

Accommodations Desired:

Prices include coffee breaks, wine and cheese parties, use of Asilomar facilities, rooms Wednesday and Thursday nights, meals from lunch Wednesday through lunch Friday. Conference attendees receive notebooks of papers presented.

Conference attendees, share a double room:

number of people _____ x \$200 = \$ _____

Attendees in single room (limited availability)

number of people _____ x \$250 = \$ _____

Non-conference guests:

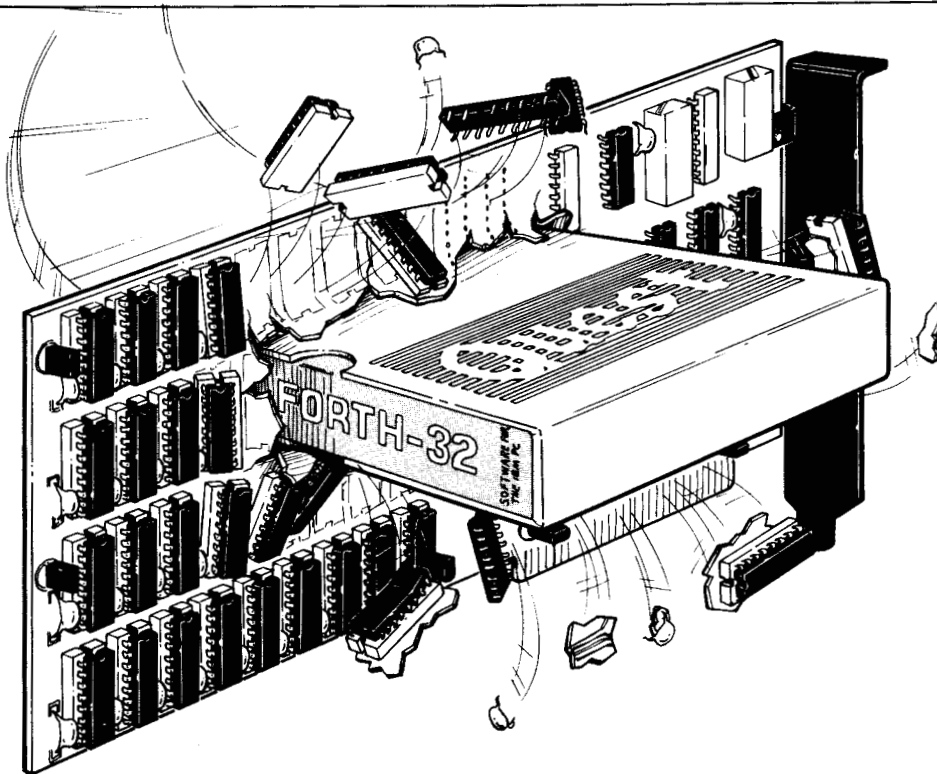
number of people _____ x \$165 = \$ _____

TOTAL ENCLOSED \$ _____

Options: Vegetarian meals?

Non-smoking roommate?

FORML, P.O. Box 51351, Palo Alto, California 94303, U.S.A.



Break Through the 64K Barrier!

FORTH-32™ lets you use up to one megabyte of memory for programming. A Complete Development System! Fully Compatible Software and 8087 Floating Point Extensions.



Quest Research, Inc.

303 Williams Ave.
Huntsville, AL 35801
(205) 533-9405

Call today toll-free or
contact a participating
Computerland store.

800-558-8088

Now available for the IBM PC, PC-XT, COMPAQ, COLUMBIA MPC,
and other PC compatibles!

IBM, COMPAQ, MPC, and FORTH-32 are trademarks of IBM, COMPAQ, Columbia Data Products, and Quest Research, respectively.

In-word Parameter Passing

Timothy Huang
Portland, Oregon

```
SCR # 312
0 \ &X CTRL-X $XX $XXXX          TDH13DEC82
1 DECIMAL
2 1 WIDTH !
3 : &X          \ put Ascii constant of X on stack
4   HERE 2+ C@ [COMPILE] LITERAL ; IMMEDIATE
5 : $XX         \ put Hex constant of XX on stack
6   FASE @ HERE 1+ NUMBER ROT BASE ! [COMPILE] LITERAL ;
7   IMMEDIATE
8 : $XXXX      \ put 16 bit Hex constant of XXXX on stack
9   [COMPILE] SXX ; IMMEDIATE
10 5 WIDTH !
11 : CTRL-X    \ put Control character of X on stack
12   HERE 6 + C@ C/L - [COMPILE] LITERAL ; IMMEDIATE
13 31 WIDTH !
14 ;S
15
```

```
SCR # 313
0 \ DUPX DROPX          TDH13DEC82
1 DECIMAL
2 3 WIDTH !
3 : DUPX              \ dup x_th item on stack
4   HERE [ WIDTH @ ] LITERAL + NUMBER DROP
5   [COMPILE] LITERAL PICK ; IMMEDIATE
6 4 WIDTH !
7 : DROPX             \ drop x_th item on stack
8   HERE [ WIDTH @ ] LITERAL + NUMBER DROP
9   [COMPILE] LITERAL ROLL DROP ; IMMEDIATE
10 31 WIDTH ! ;S
11
12
13
14
15
```

```
SCR # 314
0 \ SWAPXY              TDH13DEC82
1 DECIMAL
2 : X ( addr --- stack addr ) \ get x_th cell address
3   C@ ASCII 0 - 2* SP@ 2+ + ;
4 : Y ( addr --- stack_addr ) \ get y_th cell address
5   1+ X ;
6 4 WIDTH !
7 : SWAPXY             \ swap the x_th & y_th entries
8   HERE [ WIDTH @ 1+ ] LITERAL + DUP
9   X SWAP Y
10  VSWAP ; IMMEDIATE
11 31 WIDTH !
12 ;S
13
14
15
```

This article and short program was stimulated by one of the L.A. FIG's handouts. Screen 312 is basically a copy from that. The word **&X** functions similar to **ASCII**, except that the parameter resides within the word, *i.e.*, the **X**.

The concepts from these words are so intriguing that I decided to explore them further (Screen 313). As normal Forth will not allow the parameter to be passed within the word's name itself, by adjusting the user variable **WIDTH**, one can play the game of passing the bulk through the name of the word. Screen 399 provides two examples of how **DUPX** will duplicate the **x**th stack entry to the top of the stack. This is similar to **<x> PICK**, except the index is included in the name. **DROPX** performs similar to the combination of **<n> ROLL DROP**.

The word **SWAPXY** (Screen 314) extends the same concept one step further in that it passes two single-digit parameters. It swaps the **x**th and the **y**th stack entry. The word

VSWAP (addr1 addr2 ---)

used in line 10 will swap the contents of two addresses **<addr1>** and **<addr2>**. This word can be defined as:

```
: VSWAP ( addr1 addr2 --- )
  2DUP e >R e SWAP ! R> SWAP ! ;
```

Figure One displays some examples of usages.

```
9 8 7 6 5 4 3 2 1 CR S.
9 8 7 6 5 4 3 2 1 OK
DUP5 CR S.
9 8 7 6 5 4 3 2 1 5 OK
DROP CR S.
9 8 7 6 5 4 3 2 1 OK
DROP7 CR S.
9 8 6 5 4 3 2 1 OK
SP! OK
9 8 7 6 5 4 3 2 1 OK
SWAP37 CR S.
9 8 3 6 5 4 7 2 1 OK
SWAP29 CR S.
2 8 3 6 5 4 7 9 1 OK
SWAP15 CR S. 5 SPACES SWAP51 S.
2 8 3 6 1 4 7 9 5      2 8 3 6 5 4 7 9 1 OK
```

End Listing

Figure One

Stack-Oriented Co-Processors and Forth

*Dana Redington
Redwood City, California*

Ideally, a computer can be adapted to a wide variety of laboratory situations, provided that two conditions are met. The first condition requires using an appropriate, interactive environment. Here, Forth provides one of the best alternatives. The second condition usually requires extending software to meet current needs. Since Forth is intentionally extensible, this means molding the environment to fit the situation by enhancing the dictionary. Occasionally, increasing the vocabulary is not sufficient and more direct enhancements are needed in the form of hardware, as in floating-point computation.

This paper focuses on hardware enhancements to the Forth environment. It briefly reviews the structure of Forth, introduces co-processing, outlines the 8087 numeric processor with example words, and suggests the future of stacks in Forth.

The Structure of Forth

Forth is a unique, interactive language/environment. It is an example of what Loeliger (1981) calls a "threaded interpretive language." Additionally, Forth utilizes stacks, as do other languages (like UCSD Pascal). But, it is more than just an interactive, stack-oriented, threaded interpreter. The sum, in this case, is greater than the individual parts. Hofstadter's (1979) description of "strange loops" forming emergent phenomena is appropriate for describing what happens in Forth: "an interaction between levels in which the top level reaches back down towards the bottom level and influences it, while at the same time being itself determined by the bottom level." (p. 709).

Using this analogy, the stack resides near the bottom level of Forth. The stack is a temporary place to store and

transpose elements. Usually, stack elements are inferred to be 16-bit numbers even though other types of elements exist (e.g. character strings on a string stack or sprite planes on a graphics stack). The stack elements are stored one on top of the other where the most recently placed element on the stack is usually the first to come off, that is, a last-in-first-out stack. The stack is also a place to transform elements—using the (reverse polish) number sequence: "1 3 + ." yields "4 OK".

There are different types of stacks in Forth. The type is determined by the meaning of the stack elements—what function the elements serve. A data or parameter stack is used to store elements that usually represent data or the address of a variable. A return stack is used to store numbers that usually represent program flow-control parameters like the code field address of the next word to be executed.

Unfortunately, in Forth it becomes increasingly awkward to deal with numbers of larger sizes as in the "ripple of the carry bit" problem beyond 16 bits. The problem of larger numbers becomes more apparent on 16- and 32-bit computers. Examples include attempting to access memory beyond the 64K byte limit and dealing with numbers well beyond 16 bits as in quadruple-word arithmetic or floating-point computation. In such cases an alternative is necessary.

There are three primary alternatives to augmenting the Forth stack environment. They are, in increasing order of complexity: (1) simply devising Colon and/or Code definitions like writing floating-point routines in software and (2) adapting memory-mapped or ported hardware like a floating-point processor (such as adding a 9511 or 9512) with the necessary interface words, or (3) incorporating a co-processor specially equipped to deal with the desired

stack elements. Incorporating a co-processor is the most interesting, but seldom used, alternative.

Stack-Oriented Co-Processing

Co-processing is a special form of multi-processing. The co-processor, as a guest, lacks some of the faculties of the host processor. The guest must rely on the host for some faculties such as memory segmentation and address generation. One advantage of co-processing is that almost no overhead is incurred in setting up the guest to execute an instruction when the host and guest are working in unison. A second advantage of co-processing is that the guest can be performing a complex calculation (like raising a number to the *i*th power) while the host is performing a few "housekeeping" chores; this is referred to as an asynchronous co-processing mode. A third advantage of co-processing is that host and guest can work together in what is called maximum synchronized co-processing; the host "waits" until the instant the guest has completed a computation before continuing with the instruction stream.

Obviously, a co-processor is a device that augments a processor by extending and/or redefining the host's capability. An ideal co-processor shares the host's resources. This shared resources approach also has a severe hardware limitation: a special co-processor must exist for a specific microprocessor. And in the case of Intel's 8086, a special 8087 Numeric Data Processor (NDP) is available. (The 16081 numeric co-processor exists for National's 16000 series and Motorola is developing an NDP for the 68000).

The 8087

The 8087 is a true co-processor. It augments the register and instruction sets of the host 8086(88) microprocessor; it enhances the variety of numeric data types; and it accelerates the 8086(88)'s numeric computation capabilities. The (5Mhz) 8087 was initially

introduced in July 1980. A newer, faster (8 Mhz) 8087 is projected for January 1984. The 8087 has eight 80-bit registers, two pointers, a control register and a status register; supports seven data or number types; performs all computations on a temporary real format (80 bit); and has six principle instruction types. Detailed technical information on the 8086(88)/8087 can be found elsewhere.¹

One of the unique features of the 8087 illustrates how co-processing instructions are interpreted. In a maximum synchronized mode, the NDP interprets the instruction stream along with the host processor. The NDP remains poised until a special command sequence is detected (ESC). When the special co-processor command is read, the 8087 interprets the subsequent commands, accessing memory as necessary and executing the interpreted commands. The 8086 waits until the NDP has completed the commands before continuing. In a true co-processing fashion the 8086(88) and 8087 interpret the same instruction stream containing embedded NDP commands.

The 8087 was designed with a stack structure: "the charter of the 8087 design team was first to achieve exceptional functionality and then obtain high performance." (*iAPX manual*, p. S.3). The 8087 is a Stack-Oriented Co-processor (maybe an "SOC"?). This structure makes it compatible not only with the 8086 but also with the architecture of Forth.

Forth and a Stack-Oriented Co-processor

The stack-oriented structure of the 8087 provides an easy incorporation into an 8086 Forth environment. All that is needed is a small 8087 assembler that contains the primitive commands to communicate with the NDP (this kind of assembler is described elsewhere²). The operation of a stack-oriented co-processor is identical to the operation

of existing stacks; two examples help illustrate the simple extensibility of stack-oriented co-processing.

The first example involves a variant of integer computation. Usually, one integer is placed on the stack, then a second integer is placed on the stack and in reverse polish fashion a command is given to multiply the two numbers; for example,

```
1234 5 * .
```

would produce

```
6170 0 OK
```

My 8 Mhz 8086 Forth environment performs 3230 (integer multiply) operations per second, or 3.23 KOPS (pronounced "K-OPS").

In the co-processing case, a similar sequence is repeated. Again two numbers are placed on the stack. But, instead of issuing a `*` command, which would result in the 8086 multiplying the numbers, four additional steps must take place: (1) the first number must be moved to the co-processor's stack (the stack that physically resides within the 8087 chip; for this the word `W>F` defined in the 8087 assembler moves an integer or word from the data stack to the floating-point stack); (2) the second number must be moved to the 8087 stack, again using `W>F`; (3) the numbers must be multiplied using `F*`: (the floating-point analog of `*`; and then (4) the resulting number must be returned to the Forth data stack using `F>W` (which converts a floating-point number to a 16-bit integer and transfers it to the top of the data stack). The typed sequence

```
1234 5 W>F W>F F* F>W .
```

would produce, again

```
6170 OK
```

In this case "more is less." Even though four steps are required, the 8086 + 8087 actually performs floating-point multiplication at 29.41 KOPS. The addition of a numeric co-processor increases the computational speed for integer multiplication by about nine times.

As a side point, the NDP is capable of performing in excess of 88,8000 floating-point multiplies per second. But the observed computational speed is slower due to the overhead needed to run Forth. (What if a Forth co-processor chip existed for **DOCOL**, **SEMIS**, and **NEXT**?)

The second example of stack-oriented co-processing involves extending the Forth environment. This example is an extension of the first one. A significant difference between Forth and other languages and/or environments is the Divide-Test-Conquer approach; first divide the application into easily testable parts, test each part, and then conquer the application. Having already tested the fast variant of an integer multiply, we can now define a new word and extend the vocabulary. For added clarity let us define the new word as:

```
CODE * (W>F) (W>F)
(F*) (F>W) NEXT;
```

The words in parentheses are the primitives of their "colon-level" counterparts. `CODE *` operates on the stacks in an identical fashion as:

```
W>F WF F* F>W
```

but at 45.45 KOPS.

Additionally, after this word has been defined, any subsequent applications of `*` uses the "newer" and faster definition. The definition for integer multiply has been literally redefined. This Forth feature is rarely found in other languages. For example, attempting to redefine the integer multiply in UCSD Pascal is next to impossible.

The Future of Stack-Oriented Co-processors and Forth

Stack-oriented co-processors provide a means of extending a Forth system. For now, numeric co-processors are easily applied. Their addition not only extends computations to include

1. Good references on the 8087 include: Duncan (1982), Field (1983), Palmer, *et al.*, (1980), Rash (1981), Simington (1983).

2. I resurrected an 8086/8087 assembler written in Forth by John Bumgarner on my Seattle Computer Products 8086/8087 (Gazelle) system running at 8 Mhz. John Bumgarner and myself are

completing a draft of an article on "An Extensible Assembler for the 8087." The 8087 was placed in a copper "girdle" (only recommended for Forth artisans) to enhance heat dissipation at the increased clock frequency (thank you TZ). I have also been beta-testing a newer (non-girdled) 8 Mhz 8087 (thank you LM, JT, DC).

floatingpoint calculations but also enables microprocessor systems to rival larger mini and mainframe computers in number crunching ability. In the near future, other types of stack-oriented co-processors may become available, including string and graphics co-processors. Until a Forth processor is available, and maybe even after, a Forth co-processor chip also provides a means of extending a Forth system. In the co-processor approach, very powerful microprocessor systems can be built of various combinations of guest processors and host processors. Ultimately, each element type that requires a stack might have a custom co-processor.

The most important benefit of stack-oriented co-processors is their special ability to operate on an internal stack of predefined elements. These processors provide an effective balance of increased speed with a minimum of additional hardware and additional vocabulary. The interaction between Forth and stack-oriented co-processors forms

a strange loop where the sum is conveniently greater than the computational parts.

This paper has evolved from two earlier papers presented at the 4th Annual Forth Convention, San Jose, 9 October 1982 and at the Eighth Annual West Coast Computer Faire, San Francisco, 18-20 March 1983. Address communications to: D. Redington, Sleep Research Center, Department of Psychiatry and Behavioral Science, Stanford University School of Medicine, Stanford, California 94305.

References

Duncan, R. "Intel's 8087 Numeric Data Processor." *Dr. Dobbs's Journal*, 1982, 7(8), pp. 47-50.

Field, T. "The IBM PC and the Intel 8087 Co-processor, Part 1: Overview and Floating-Point Assembly-Language Support." *Byte*, 8(8), 1983, pp. 331-374.

Hofstadter, D. R. *Godel, Escher, Bach: an eternal golden braid*. New York: Basic Books. 1979.

Loeliger, R. G. *Threaded Interpretive Languages*. Peterborough, NH: Byte Books. 1981.

Palmer J., Nave R., Wymore C., Koehler R., & McMinn C. "Making Mainframe Mathematics Accessible to Microcomputers." *Electronics*, 8 May, 1980, pp. 114-121.

Rash B., "Application Note AP-113 — Getting Started with Numeric Data Processors." Intel Corporation, February 1981.

Simington, R. B. "The Intel 8087 Numerics Processor Extension." *Byte*, 8(4), 1983, pp. 154-172.

iAPX 86,88 USER'S Manual. Intel Corporation, 1981.

1 proFORTH COMPILER 8080/8085, Z80 VERSIONS

- SUPPORTS DEVELOPMENT FOR DEDICATED APPLICATIONS
- INTERACTIVELY TEST HEADERLESS CODE
- IN-PLACE COMPILATION OF ROMABLE TARGET CODE
- MULTIPLE, PURGABLE DICTIONARIES
- FORTH-79 SUPERSET
- AVAILABLE NOW FOR TEKTRONIX DEVELOPMENT SYSTEMS — \$2250

2 MICROPROCESSOR-BASED PRODUCT DESIGN

- SOFTWARE ENGINEERING
- DESIGN STUDIES — COST ANALYSIS
- ELECTRONICS AND PRINTED CIRCUIT DESIGN
- PROTOTYPE FABRICATION AND TEST
- REAL-TIME ASSEMBLY LANGUAGE /proFORTH
- MULTITASKING
- DIVERSIFIED STAFF

MICROSYSTEMS, INC.

(213) 577-1471

2500 E. FOOTHILL BLVD., SUITE 102, PASADENA, CALIFORNIA 91107

Code and Colon Compatibility

David Held
Hermosa Beach, California

I recently developed an approach to a problem which may be of interest to others. I was developing a communications application in Forth-79 which would require some code definitions due to speed requirements. For the development process, however, I preferred to begin with colon definitions, planning to convert progressively higher-level words into code as the work progressed. Thus, I faced the problem of creating code definitions for words that might be called either as subroutines from other code words, or as Forth-compatible words from colon definitions.

The critical difference between the two is that a machine-language subroutine should end by returning to its caller (pop the stack for the caller's address), whereas a Forth code definition ends by jumping to **NEXT**. To resolve the dilemma, I used the techniques illustrated in Figure One by 8080 machine language.

These words would permit me to make calls to any machine-language subroutine (such as in my system's monitor); for example,

```
: SCROLL F010 CALL ;
```

would write **F010** over the zero in the definition of (**CALL**), and would then

```
CODE (CALL)      ( a word which calls a subroutine, then jumps to NEXT )
  CD C, 0        ( equivalent to CALL 0 )
  NEXT JMP,      ( equivalent to JMP NEXT )
  END-CODE

' (CALL) 1+ CONSTANT CALL-ADDR      ( a constant containing the address )
                                           ( of the zero in the above definition. )

: CALL      ( addr --- ) ( a word which calls the subroutine at addr )
  CALL-ADDR ,      ( write the desired subroutine address )
                ( over the zero in definition of (CALL) )
  (CALL) ;        ( and execute (CALL) )
```

Figure One

```
: SUBROUTINE      ( create a code subroutine useable either from colon )
                  ( or code definitions. )

[COMPILE] ASSEMBLER ( invoke assembler vocabulary )
CREATE           ( create a header for the subroutine )
DOES> CALL ;     ( this happens when subroutines so )
                  ( defined are invoked from FORTH )
```

Figure Two

execute (**CALL**). Thus the monitor subroutine at (**F010**) would be executed, followed by a jump to **NEXT**.

Now we want to create subroutines that can be used either from colon definitions or code words. Here is a defining word which defines such subroutines. When invoked from colon definitions, the run-time behavior is similar to **CALL**, above. To use the subroutine from code definitions, merely "tick" its address and **CALL** it in assembly language. Figure Two shows the defining word **SUBROUTINE**:

For example, this word might be used to define subroutine **SUB1**, as follows:

```
SUBROUTINE SUB1
80 A MV1
C010 STA
RET
```

Now we can invoke **SUB1** from Forth, as follows:

```
: TEST1 SUB1 ;
```

Or, from a machine-language word, as follows:

```
CODE TEST2
' SUB1 CALL,
NEXT JMP,
END-CODE
```

The advantage we gained by all this manipulation is that the definition of **SUB1** is unchanged, whether it is used by a colon or code word.


Inner Access holds
the key to your
software solutions



When in-house staff can't solve the problem, make us a part of your team. As specialists in custom designed software, we have the know-how to handle your application from start to finish.

Call us for some straight talk about:

- Process Control
- Automated Design
- Database Management
- System Software & Utilities
- Engineering
- Scientific Applications
- Turn Key Systems

 Inner Access Corporation
P.O. Box 888, Belmont, CA 94002
PHONE (415) 591-8295

CORDIC Algorithm Revisited

Dave Freese

Cape May Court House, New Jersey

I am employed as a senior engineering analyst and routinely use PL/I, Pascal, Fortran and BASIC at my place of employment. Until recently, Forth was just for hacking around at home, but it may just prove to be the answer to a "real" estate problem at work. I am developing special purpose wind speed/direction instruments for the U.S. Navy and have been frustrated by the "fat" code produced by all of the compilers at my disposal. None of the compilers have the option of pruning the object code by removing unwanted support code. When you are downloading the resultant code to ROMs, a trade-off must be made between code size and speed. Forth may just let me have the best of both worlds.

Which brings me to the subject of source code. The Volume IV, Number 1 issue of *Forth Dimensions* has found a permanent place on my desk. It contains some of the best material on fixed-point arithmetic that is available (at least with regard to Forth). The article on vector rotation using the CORDIC algorithm was particularly useful to me, as that type of conversion is routinely performed in wind speed/direction computation. The original code did not, however, meet my expectations with regard to accuracy. In particular, it failed to return correct values for rotations of 0, 45, 90 and 180 degrees. The accompanying listing of a double-precision version of the algorithm will provide the needed accuracy. It converts all input into double-precision numbers with the binary point between bits 15 and 16 (*i.e.*, the upper half of a number represents the integer part and the lower half represents the fractional part).

This was written for Z80 Forth by Laboratory Microsystems. This interpreter allows double number literals in colon definitions. Modify these entries for FIG-Forth or other interpreters which do not allow this extension. The conversion is fast due to the machine code divide-by-factor-of-two, 2SRA routine.

```
Screen # 27                                crc = 25114
0 ( CORDIC ALGORITHM -- words for machine code )
1
2 HEX
3
4 ( create header with CFA pointing to body of word )
5 : :CODE ( -- ) BASE @ HEX CREATE ;
6
7 ( terminate body of word with a jump to NEXT )
8 ( jp next )
9 : ;NEXT ( -- ) @C3 C, NEXT-LINK , SMUDGE
10 BASE ! ;
11
12 DECIMAL
13
14
15 -->
```

```
Screen # 28                                crc = 5781
0 ( CORDIC ALGORITHM -- double number words )
1 ( 2DUP 2SWAP 2DROP D+ D- 2@ 2! previously defined )
2 : 2OVER ( d1 d2 -- d1 d2 d1 ) >R >R 2DUP R> R> 2SWAP ;
3 : 2ROT ( d1 d2 d3 -- d2 d3 d1 ) >R >R 2SWAP R> R> 2SWAP ;
4 : 2VAR ( -- ) <BUILDS 0 0 , , DOES> ;
5 : 2CON ( -- ) <BUILDS , , DOES> 2@ ;
6 : D< ROT 2DUP = IF ROT ROT DMINUS D+ 0< ELSE
7 SWAP < SWAP DROP THEN SWAP DROP ;
8 : D* ( d1 d2 -- d3 ) OVER 5 PICK U* 6 ROLL 4 ROLL * +
9 2SWAP * + ;
10
11 :CODE 2SRA ( d n -- d/2^n ) @E1 C, @7D C, @B7 C,
12 @CA C, NEXT-LINK , @E1 C, @CB C, @2C C, @CB C,
13 @1D C, @E3 C, @CB C, @1C C, @CB C, @1D C, @E3 C,
14 @3D C, @20 C, @E3 C, @E5 C, ;NEXT
15 -->
```

```
Screen # 29                                crc = 51411
0 ( CORDIC ALGORITHM ) -->
1 Zilog mnemonics for definition of 2SRA
2
3 POP HL $1: SRA H
4 LD A,L RR L
5 OR A,A EX (SP),HL
6 JP Z,NEXT RR H
7 POP HL RR L
8 EX (SP),HL
9 cont'd next column DEC A
10 JR NZ,$1
11 PUSH HL
12 JP NEXT
13
14
15
```



```

Screen # 30                                crc = 16435
0 ( CORDIC ALGORITHM )
1 ( ALPHA[i] = 65536 * 32768 * arctan[1/2^i] / Pi )
2 2VAR ALPHAS -4 ALLOT
3 536870912. , , 316933406. , , 167458907. , , 85004756. , ,
4 42667331. , , 21354465. , , 10679838. , , 5340245. , ,
5 2670163. , , 1335087. , , 667544. , , 333772. , ,
6 166886. , , 83443. , ,
7 ( convert double to single with round up )
8 : D->S 32768 0 D+ SWAP DROP ;
9 : RVSUB1 >R 2ROT 2ROT 2OVER 2OVER R> 2SRA ;
10 : RVSUB2 >R 2ROT 2ROT 2SWAP R> 2SRA ;
11 : RVSUB3 >R 2ROT R> 4 * ALPHAS + 20 ;
12 : *KN ( n -- d = n * 65536 * 0.60725293 )
13 256 /MOD SWAP >R S->D 10188014. D*
14 R> S->D 39797. D* D+ ;
15 -->

```

```

Screen # 31                                crc = 23246
0 ( CORDIC ALGORITHM )
1 : ROTVECTOR ( n.y-old n.x-old n.ang -- n.y-new n.x-new )
2 >R ( save angle )
3 >R ( save n.x ) *KN ( convert n.y )
4 R> *KN ( convert n.x ) 0 R> ( retrieve angle -> double )
5 2DUP 0 0 D< IF 0 16384 D+ 2ROT 2ROT DMINUS 2SWAP 2ROT
6 ELSE 0 16384 D- 2ROT 2ROT 2SWAP DMINUS 2ROT THEN
7 14 0 DO
8 2DUP 0 0 D< IF
9 I RVSUB1 D- I RVSUB2 D+ I RVSUB3 D+ ELSE
10 I RVSUB1 D+ I RVSUB2 D- I RVSUB3 D- THEN
11 LOOP 2DROP ( drop d.angle )
12 D->S >R D->S R> ; -->
13
14
15

```

```

Screen # 32                                crc = 40282
0 ( CORDIC ALGORITHM )
1 ( single precision angular conversions )
2 : PIRADIANS MINUS 32768 ROT ROT */ ;
3 : DEGREES 180 PIRADIANS ;
4 : POLAR-> ( rad ang -- y x ) 0 ROT ROT ROTVECTOR ;
5
6
7
8
9
10
11
12
13
14
15 ;S

```

End Listing

FOR 8080, Z80, 8086*, 68000*

MULTIUSER MULTITASKING

A professional quality full feature
FORTH system at a micro price.

TaskFORTH™

Single, double, triple,
quadruple and floating point
math, trigonometric functions

Case statements

Interactive debugger

Novice Programmer
Protection Package™

Multiple thread dictionary

System date/calender clock

Hierarchical file system

Screen and serial editor

Inter-task communications

Unlimited number of tasks

Starting FORTH, FORTH-79
and FORTH-83† compatible

Graphics support

TaskFORTH is the FORTH
system you would write,
if you had the time . . .

ALL included for just \$395
(plus applicable taxes)

Available for CP/M, Northstar DOS,
Micropolis and Stand-alone.

Visa & MC Accepted

* Available soon
† When standard is approved

CP/M is a trademark of Digital Research
TaskFORTH is a reg. trademark of Shaw Labs, Ltd.

Single user, single computer license agreement
is required.

SHAW LABORATORIES, LIMITED
24301 Southland Drive, Suite 216
Hayward, California 94545
(415) 276-5953

Forth-83 Standard

Robert L. Smith
Sunnyvale, California

As many readers are aware, the Forth-83 Standard has been approved by the Forth Standards Team. By the time you read this, copies of the standard should be available from the Institute for Applied Forth Research, MicroMotion, or Mountain View Press. The majority of members of the Forth Standards Team are vendors or potential vendors of Forth systems and applications. The Forth-83 Standard represents a substantial input from the team members, the referees, and the Forth community. Literally hundreds of proposals were received and examined. In contrast to the past, there were two major meetings of the Standards Team as a whole, and very many meetings of the referees. The result is a document of substantial quality. The team rules require a two-thirds affirmative vote of the members to accept the new standard. The actual vote as of the time of this writing is twenty-two "yes" votes, one "no" vote, and three votes not received. We can see that the vote was quite decisively in favor of the new standard. In my opinion, the new standard offers a significant improvement over previous Forth standards.

Like all standards, Forth-83 is the result of many compromises and, therefore, not all readers will agree on the desirability of some of the features. It should be pointed out that the Forth Standards Team is not forcing anyone to adhere to the standard. In the forward to the standard, the following sentence appears: "A programmer or vendor may choose to strictly adhere with the standard, but the choice to deviate is acknowledged as beneficial and sometimes necessary." Certainly, if one has programs which work on an older standard or a non-standard system, there is no requirement that the old system be thrown away and the programs rewritten just because a new standard exists.

Let us briefly review some of the differences from the previous standard

which may affect a more general acceptance of Forth-83. Most of the issues have been previously aired in this column for the purpose of information and to encourage public input.

The new **DO-LOOP** is somewhat different from previous **DO-LOOPS**. Most people now seem to prefer the new, circular-arithmetic **DO-LOOP**. Briefly, the advantages are (1) the index **I** now has a full 65K range, (2) there is no longer a need to have a separate **/LOOP** for unsigned indices, and (3) in most cases the new loop is faster than most older ones. A few vendors would prefer that

```
0 0 DO ... LOOP
```

would cause a null result. It appears that if that result is desirable, it could be obtained by using the new System Word Set and defining the desired function with a different name. Note that old code which used the construct

```
0 0 DO
```

would be incompatible in either case. A closely related issue is that of the new version of **LEAVE** which causes control to transfer to the end of the loop. There is an implementation issue here. There is not adequate space to discuss the issue thoroughly, but one vendor would prefer to have either a more complicated form (called **LEAVES**) or, alternatively, to allow only one occurrence of **LEAVE** within a given loop.

The default value of "true" for comparison operators now returns all bits set rather than just the low-order bit. In most cases, a comparison is followed by a test for non-zero, such as the word **IF**. In that case there will be no difference. If old code uses comparisons in conjunction with arithmetic operations, then some change will be required to work under the new standard. The simplest change is to follow the comparison with a negation operator. The new default value for "true" should be somewhat more useful than the old value. A related side

benefit is that the word **NOT** is now available to mean "take the one's complement," whereas previously it was synonymous with **0=**. In most cases it is compatible with previous usage.

Historically, division in Forth has varied from system to system. According to Charles Moore, if a machine had a hardware divide, then its characteristics determined the division result. If division had to be done in software as with, say, the 8080, then floored division was usually chosen. However, only positive denominators were generally considered. In some cases **/** took signed arguments and **/MOD** took unsigned arguments. In Forth-83, the result of the **/** operation is the mathematical floor of the real number quotient. Alternatively, one may say that the quotient is more useful than the 79-Standard version. For example, one can readily perform an arithmetic right shift by dividing by an appropriate power of two. In hexadecimal arithmetic,

```
8712 100 /MOD
```

will yield a quotient of FF87 and a modulus of 12, so that the original number is readily split into 8-bit components. Under 79-Standard, the operation would yield FF88 with a remainder of FF12. A nice result is that now the right-shift operator **2/** is identical to **2 /**. In my opinion, the enhanced utility of the Forth-83 quotient and modulus function outweighs the disadvantage that some older code may need to be somewhat modified when negative arguments are employed.

In many cases, previous ambiguities have been resolved or clarified. There is at least one word, **J**, which is ambiguous in certain rare cases. It is important to realize that the new standard has not changed the meaning of this word from previous standards. The problem arises from alternative

(Continued on page 30)

Forth-83: A Minority View

Glenn S. Tenney
Belmont, California

Having participated as a member of the Forth Standards Team (FST) and as a referee of the 83-Standard, I am strongly in favor of a new Forth Standard. Despite the many weeks I devoted to this, my professional conscience did not allow me to vote to accept the 83-Standard. For these reasons, I was asked to document my concerns as well as those of others. As you read this, please remember that many of the following are the concerns of others and do not necessarily represent my own views.

The Forth 83-Standard has recently been accepted and is now being published. Many people have been expressing varying degrees of concern over this new standard. After discarding a certain amount of gripes, these concerns fall into one of four categories: minor and non-technical;

incompatibilities to the prior standard or existing systems; specific technical points; and general philosophical points. This is an overview of some of those concerns.

The new standard is almost completely reworded. This was done to make it more readable, yet this generated some fairly minor concerns, the most obvious being the process of locating underlying technical aspects that were actually changed. In fact, the new wording actually changed the published version of the standard so that **BASE** and **D<** reflect a technical change from the 79-Standard.

The new standard is incompatible with prior systems partly because the function of many words changed yet the word names remained the same. These functional changes range from obscure to obvious. This shows itself in the following ways:

- **PICK** and **ROLL** were changed, with no strong technical reason, from 1 origin to 0 origin, totally invalidating prior source code.

- Many changes were made, albeit for valid technical reasons, which are not always incompatible. These are the most dangerous, since when the incompatibility crops up it is often buried inside a previously functional definition.

The following changes affect many standard words:

- * The true flag (-1) returned from standard words provides some extra power, but can be incompatible if the flag is used in calculations.

- * Division is now floored towards negative infinity rather than towards zero.

83-Standard does not have state-smart required words. This seems to have grown from the problems using

FORTH-79

Ver. 2 For your APPLE II/II+

The complete professional software system, that meets ALL provisions of the FORTH-79 Standard (adopted Oct. 1980). Compare the many advanced features of FORTH-79 with the FORTH you are now using, or plan to buy!

FEATURES	OURS	OTHERS
79-Standard system gives source portability.	YES	_____
Professionally written tutorial & user manual	200 PG.	_____
Screen editor with user-definable controls.	YES	_____
Macro-assembler with local labels.	YES	_____
Virtual memory.	YES	_____
Both 13 & 16-sector format.	YES	_____
Multiple disk drives.	YES	_____
Double-number Standard & String extensions.	YES	_____
Upper/lower case keyboard input.	YES	_____
LO-Res graphics.	YES	_____
80 column display capability	YES	_____
Z-80 CP/M Ver. 2.x & Northstar also available	YES	_____
Affordable!	\$99.95	_____
Low cost enhancement option:		
Hi-Res turtle-graphics.	YES	_____
Floating-point mathematics.	YES	_____
Powerful package with own manual.		
50 functions in all.		
AM9511 compatible.		
FORTH-79 V.2 (requires 48K & 1 disk drive)		\$ 99.95
ENHANCEMENT PACKAGE FOR V.2		
Floating point & Hi-Res turtle-graphics		\$ 49.95
COMBINATION PACKAGE		\$139.95
(CA res. add 6% tax; COD accepted)		

MicroMotion

12077 Wilshire Blvd. # 506
L.A., CA 90025 (213) 821-4340
Specify APPLE, CP/M or Northstar
Dealer inquiries invited.



FORTH-79

Version 2 For Z-80, CP/M (1.4 & 2.x),
& NorthStar DOS Users

The complete professional software system, that meets ALL provisions of the FORTH-79 Standard (adopted Oct. 1980). Compare the many advanced features of FORTH-79 with the FORTH you are now using, or plan to buy!

FEATURES	OURS	OTHERS
79-Standard system gives source portability.	YES	_____
Professionally written tutorial & user manual.	200 PG.	_____
Screen editor with user-definable controls.	YES	_____
Macro-assembler with local labels.	YES	_____
Virtual memory.	YES	_____
BDOS, BIOS & console control functions (CP/M).	YES	_____
FORTH screen files use standard resident file format.	YES	_____
Double-number Standard & String extensions.	YES	_____
Upper/lower case keyboard input.	YES	_____
APPLE II/II+ version also available.	YES	_____
Affordable!	\$99.95	_____
Low cost enhancement options:		
Floating-point mathematics	YES	_____
Tutorial reference manual		
50 functions (AM9511 compatible format)		
Hi-Res turtle-graphics (NoStar Adv. only)	YES	_____
FORTH-79 V.2 (requires CP/M Ver. 2.x).		\$99.95
ENHANCEMENT PACKAGE FOR V.2:		
Floating point		\$ 49.95
COMBINATION PACKAGE (Base & Floating point)		\$139.95
(advantage users add \$49.95 for Hi-Res)		
(CA. res. add 6% tax; COD & dealer inquiries welcome)		

MicroMotion

12077 Wilshire Blvd. # 506
L.A., CA 90025 (213) 821-4340
Specify APPLE, CP/M or Northstar
Dealer inquiries invited.



polyFORTH II

The Operating System and Programming Language designed especially for

REAL-TIME APPLICATIONS

- Robotics
 - Instrumentation
 - Process Control
 - Graphics
- ... and many more.

polyFORTH II has the high-performance features you need to slash development time by months:

POWER

All the programming tools you need — multiprogrammed OS, FORTH compiler and assembler, editor, over 400 primitives and debugging aids — resident and ready to use.

SPEED

3-5 times faster than Pascal, 20 times faster than Basic, with a resident assembler for time-critical functions.

MULTITASKING/MULTI-USER

Supports any number of tasks. Even the smallest systems may have two or more programmers coding and testing interactively.

COMPACT CODE

Entire development system resident in under 12K. ROMable applications can run under 1K. Large applications up to 10 times smaller than with other techniques.

SUPPORT

On-line interactive documentation, over a thousand pages of manuals, FORTH Programming Courses, and the FORTH, Inc. Hot Line plus Contract Programming and Consulting Services.

Available for most popular minis and micros. From FORTH, Inc., the inventors of FORTH, serving professional FORTH programmers for ten years.

FORTH, Inc.

2309 Pacific Coast Hwy.
Hermosa Beach
CA 90254

(213) 372-8493
TWX 910-344-6408
(FORTH INC HMBH)



'(tick). In 79-Standard, **FIND** wasn't of much use so most people used the state-smart '(tick). This led to problems, since some state-smart words must be used cautiously when compiled within a colon definition. The solution adopted was to make all words state-dumb and to add words to handle some of the lost functions. **FIND** was later made useful, yet state-dumb words remain.

A side effect of the state-dumb "sweep" is that some words were made state-dumb that would have been better left state-smart regardless of the '(tick) decision. For example, there is no reason why **ABORT**" should be state-dumb. The system-dependent functions associated with **ABORT**" only appear within **ABORT**". **ABORT**" could have been of great use outside of a colon definition, yet that has now been precluded.

Some current systems use mono-addressing (e.g. CFA or PFA), yet the standard now dictates dual-addressing (e.g. CFA and PFA). This tradeoff favors an easier implementation and was the subject of many hours of discussion. Some of the original dissenters still believe that a mono-addressing system is easier to use.

Disk (mass storage) I/O has been changed to disallow altering the data within a block buffer unless **UPDATE** is also used. Even if **EMPTY-BUFFERS** (which is no longer a required word) were used, the alterations might still be written to disk. Combined with the multi-tasking implications, **BUFFER** can no longer be used for a temporary scratch or data collection area. Although this codifies good programming practice, there are valid reasons why the standard should not be so restrictive.

A minor but grating problem is that the FST itself is always trying to allow a wide variety of actual or possible implementations. This has caused some problems and ambiguities within the standard. This is more an underlying political problem than a technical one, but the results affect the standard technically:

- The] (right-bracket) definition is ambiguous. Different systems are im-

plemented such that the same supposedly transportable standard program produces different results on the different systems.

- Vocabularies have always been a problem. The standard attempts to resolve this issue, but actually leaves some gaping holes. Because search orders are not definitively stated, a standard program will have some difficulty using vocabularies and search orders.

Consumers will have great difficulty determining if a system is actually Forth-83. Although this has always been a problem, it will be worse this time because 79-Standard systems have been marketed. Implementors will have difficulty determining or choosing to make some of the subtle changes from 79-Standard.

The standard should codify existing practice. Instead, new concepts have been accepted without a complete test-bed existing with which to gain experience. In most cases, these new concepts are going to be proven correct, but what about the ones that aren't? Where will we be in four or five years if next month we find that one of these insufficiently tested concepts causes a problem?

It is difficult to judge the impact of some of these technical or philosophical concerns. Time and experience will tell whether the decisions made for Forth-83 were correct. Although it is difficult, we must temper our frenzy for having a new standard with a striving for reasonable perfection of the standard.

Letters (Continued from page 4)

8080 Conditions

Dear FIG:

John Cassaday left conditional calls out of his 8080 assembler, which he published in *Forth Dimensions* (Vol. III, No. 6). I have written a single word that adds all the 8080 conditional calls to his assembler:

```
: IFCALL SP@ 0D TOGGLE C, , ;
```

The stack for proper use is:

```
<call-addr> <conditional> ---
```

Where <conditional> is any one of John's words **0=**, **CS**, **PE**, or **0<**, and an optional **NOT**. John's conditional words leave an 8080 conditional jump op-code on the stack. My word toggles bits in it to make it into the appropriate conditional call op-code. The call-address must be next down on the stack. A **LABEL** word can be used to define the entry point to the subroutine.

Using similar techniques, I have also written words for conditional return from subroutine and conditional jump:

```
: IFRET SP@ 0A TOGGLE C, ;  
: IFJMP SP@ 08 TOGGLE C, , ;
```

IFJMP needs a jump address on the stack, under the conditional, but **IFRET**, of course, needs only a conditional on the stack.

I think these words are yet more examples of an amazing property of Forth: the solution to a problem is usually less complicated than you think it will be!

Sincerely,

Paul E. Condon
6219 Rockwell St.
Oakland, CA 94618

Yet Another Case Statement

The main feature of this **CASE** is the technique used. It is presented as an educational example of the power of Forth. In this example, **CASE** is a defining word which creates **TEST**. If you'll notice in the definition of **EXECUTES**, it looks like I'm jumping into the PFA instead of the CFA. This is because I am.

I was going to explain this, but after giving it some thought, I decided that it would be more beneficial to our fellow Figgers to present it as a puzzle. By the time you figure it out, you will understand **DOES**, the return stack, and the Forth compiler inside out.

As far as how this **CASE** compares to other **CASE**s: it compiles much smaller than most, it branches to **ENDCASE** upon finding a match, and executes about as fast. It will only execute one word after **EXECUTES** and that must be a word created in a colon definition. I don't use this **CASE** myself, nor do I think it best. It's just an example of the unusual. If you figure it out, it will help you understand Charles Moore's **BASIC** compiler (Vol. III, No. 6), which is even trickier. Good luck!

Marc Perkel
Perkel Software Systems
1636 N. Sherman
Springfield, MO 65803

Searching for Graves

Dear FIG,

I read with interest the letter from

Nick Francesco in *Forth Dimensions* (Vol. IV, No. 6). I share his feeling regarding the use of standard DOS files. I currently am using **MicroMotion's Forth**, which certainly is fine as far as the normal Forth operating systems go.

Mr. Francesco mentioned that William Graves' **Forth II** for the Apple II uses Apple's **DOS**. Looking through your section on system vendors, I could not find anything that looked like a potential Graves source. If you could supply me with more information regarding **Graves Forth II**, I would appreciate it. Or, if you do not have the information at hand, perhaps you could forward this letter to Mr. Francesco.

Thank you very much.
Sincerely,

James W. Patton
737 W. Davies Way
Littleton, CO 80120

(Continued)

```
SCR # B  
0 ( CASE EXAMPLE DEMONSTRATING FANCY BRANCHING, DOES, AND  
1 THE POWER OF THE FORTH COMPILER )  
2  
3 ; SET-RETURN R> DUP @ >R 2+ >R ;  
4 ; IFJUMP OVER = IF DROP R> @ >R ELSE R> 2+ >R THEN ;  
5 ; CASE CREATE J COMPILE SET-RETURN HERE 0 , DOES> >R ;  
6 ; EXECUTES COMPILE IFJUMP FIND 2+ , ; IMMEDIATE  
7 ; ENDCASE COMPILE DROP HERE SWAP !  
8 COMPILE EXIT [COMPILED] C ; IMMEDIATE  
9  
A ( ALL WORDS ARE 79-STANDARD, FIND RETURNS THE CFA OF THE NEXT  
B WORD IN THE INPUT STREAM. )  
C  
D  
E  
F  
  
SCR # C  
0 ( CASE EXAMPLE )  
1 ; PRINT-1 ." ONE " ;  
2 ; PRINT-2 ." TWO " ;  
3 ; PRINT-3 ." THREE " ;  
4  
5 CASE TEST  
6 1 EXECUTES PRINT-1  
7 2 EXECUTES PRINT-2  
8 3 EXECUTES PRINT-3  
9 ENDCASE  
A  
B ( 1 TEST ONE OK  
C 2 TEST TWO OK  
D 3 TEST THREE OK  
E 4 TEST OK )  
F
```

CASE Statement

Forth Family Foiled

Dear Editor,

First, I heartily endorse your screen CRC words (*Forth Dimensions*, Vol. IV, No. 3).

Second, I am looking for a nice Forth for Apple III CP/M. I am aware of several systems for Apple II, but these require emulation mode, and this hampers operations significantly. The hard disk cannot be accessed with them and time is wasted loading the emulator.

Yesterday I told my two brothers-in-law, who already own Apple IIIs with CP/M cards, "No problem, there are lots of good Forths out there." Today I spent an hour on the phone calling Forth vendors and struck out. Is anyone out there catering to deluxe Apple III owners? They want a complete Forth system with full source code, some meta compiler, a screen editor, maybe strings and floating

point, and perhaps some brief customizing documentation.

Love Forth,

Gary Nemeth
2727 Hampton Rd.
Rocky River, OH 44116

ENCLOSE Encounters of the Second Kind

Dear Editor:

Reference *ENCLOSE Encounters*, in the Technotes section of Vol. V, No. 1. A line of code near the bottom of page thirty-four was omitted. It should have read,

HEX FIRST 2+ 400 BLANKS

(blanks a block buffer)

FIRST 2+ BL ENCLOSE

Thank you again for publishing the note.

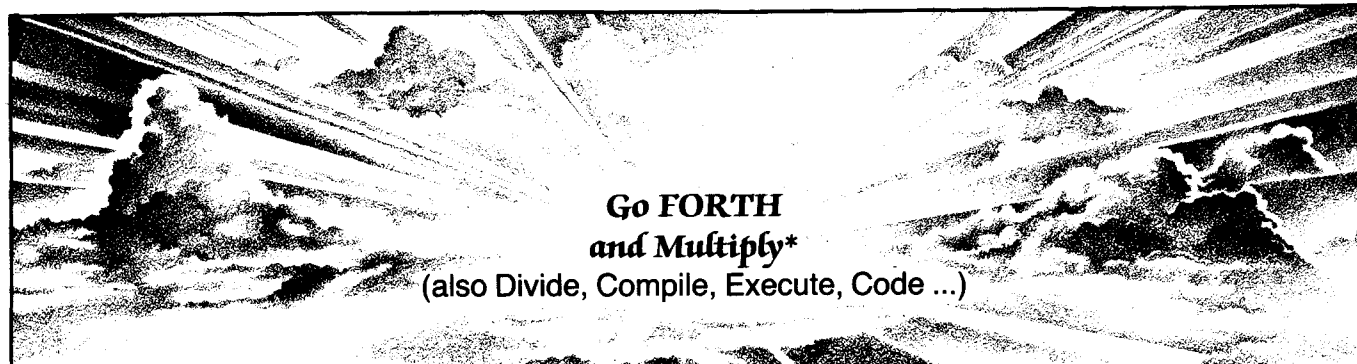
Sincerely yours,

Nicholas L. Pappas, Ph.D.
1201 Bryant St.
Palo Alto, CA 94301

(Continued from page 26)

techniques for compiling, and may occur when J is compiled into a definition. This subject will certainly be a topic of discussion for the next meeting of the Forth Standards Team. Again, the definition of J has not changed, and therefore should not be a reason to reject the new standard.

By adopting the new standard, the Forth Standards Team accepts Forth-83 as the current official FST standard, superceding all prior standards. It is important that we have a standard for writing transportable application code, as a basis for writing books and documents, and for teaching and communication.



Go FORTH
and Multiply*
(also Divide, Compile, Execute, Code ...)

Have You Gotten The Word Yet?


Companies such as IBM, Atari, Varian, Hewlett Packard, Dyan and Memorex are now using FORTH for a number of applications. If you are concerned about efficiency and transportability, then FORTH is a language you should learn.

FORTH Fundamentals	\$395.00
Advanced Systems & Tools	\$495.00

(For further information, please send for our complete FORTH workshop catalogue).

Join the FORTH Revolution!

- Intensive 5-day workshops
- Small classes
- Experienced professionals
- On-site classes by special arrangement

 **Inner Access Corporation**
P.O. Box 888, Belmont, CA 94002
(415) 591-8295

Meta Compiling III

*Henry Laxen
Berkeley, California*

Last time we talked about how to implement **CODE** words in the meta compiler, and saw how such words must operate in order to make meta : definitions work. We also saw how to define a symbol table for the definitions that are created during meta compilation by using the existing vocabulary structure. We also looked at how to create headers in the target address space. If any of these concepts are unfamiliar to you, I suggest you reread the previous two articles in this series, which discuss them in detail.

I would now like to talk about a few of the subtle issues that come up during meta compilation that must be handled by some means or another. Some of the subtle issues are how to handle forward references, and how immediate words such as `.` are handled. Other similar issues arise, but we must leave some questions unanswered so that the reader can experience the joy of discovery.

The issue of forward references during meta compiling has, for some unknown reason, become almost a religious issue. The regular Forth interpreter treats forward references as an error condition, which has its pros and cons. Fortunately, it is almost always possible to write your Forth application in such a way that you can avoid forward references, hence one branch of the religion considers the problem solved, namely, don't use forward references. Unfortunately, in the meta compiling process, forward references are unavoidable, and we must develop a technique to handle them. Before I discuss a few solutions, I would like to present my view of the forward reference issue. The use of forward references is not sinful, immoral, illegal, or fattening. It should be discouraged but not banned. The problem that arises with forward references is that you can get yourself into big trouble. It destroys the bottom-up nature of Forth, and can cause you to retest previously working words because

they make use of a forward reference which has changed. It also decreases the usefulness of program listings, if you are never sure of which way to turn the pages when you encounter an unfamiliar word.

Forward references also complicate the compiler, since it now must handle another class of objects (other than previously defined words and numbers). Most threatening, however, is that if forward references are abused, you can wind up with totally undecipherable spaghetti code. Just look at almost any Fortran program larger than 100 lines and written by a physicist, and you will see what I mean. The case for forward references is that sometimes you must have them. For example, if you are using recursion, and word **A** calls word **B** which calls word **A**, I am afraid a forward reference is somewhat unavoidable; if recursion is the natural solution to your problem, it would be silly not to use it. Also, error conditions are often more easily handled if forward references are allowed. You will often want a fatal error, which could occur at a relatively low level in your program, to call, say, the main menu routine, which obviously occurs at a very high level. That is the case in Forth, where **ABORT**, which is used at a very low level, calls the Forth interpreter, which is defined at a very high level.

Enough religion, let's take a look at some techniques for handling forward references during meta compilation. The simplest method to implement (and the hardest to control) is to simply use a place-holding word, and then patch it later when the resolving forward reference word is defined. Normally this word is called **GAP** (and it behaves like a comment, skipping the following text until the next `)` and simply compiling a zero into the target image. The intervening text is usually the name of the word which will later be patched into this location. The problem with this approach is that you have

to index some number of bytes, depending on the location of the gap, into the word that was defined, and patch it with another value. This approach is very inflexible and error prone, since if you ever change the definition in which the gap occurs, you must also change the place where it gets patched in a corresponding manner. There is no intelligence required, just conscientious effort, something humans are not well equipped for.

Another approach is to explicitly declare a forward reference before it takes place, and then resolve it somehow later when its target address is known. This is the Pascal approach, and is a pretty good compromise. At least you no longer have to count bytes into a word and hot patch it later. You can simply name the forward referenced word and define a mechanism that resolves it. This approach also allows you to have multiple forward references by linking them into a chain, and resolving the entire chain once the target address is known.

Finally, the last approach I will mention is that of handling forward references on the fly. I do not mean to imply that there are only three ways of doing this; there are many more, but three is enough for now. In order to handle forward references on the fly, we must modify the meta compiler's compiler. Instead of issuing an error message when an undefined word turns out not to be a number, we must define the word in question and remember the fact that it is a forward reference. Basically, all this entails is to change the compile loop to decide upon one of three cases instead of only two. Case one is that the word to be compiled already exists, in which case we simply compile it by executing it and letting it compile itself. Case two is that the word is a number in the current base, in which case we compile the code field for literal, followed by the value of the literal. Case three is that the word to be compiled is not already defined and is

not a number, hence it must be a forward reference. In this case we must create an entry for it in the symbol table of forward references, compile a gap in the word currently being defined, and set up the run time of the forward reference to either link itself into a chain if it is not already resolved, or to compile itself if it is already resolved. Thus, forward references become basically transparent, except that they must be resolved somehow. This resolution can either be automatic as the word is actually defined, or explicit, requiring you to issue commands that will cause the resolution. Personally, I prefer the explicit method, since I am afraid of things happening behind my back, and it slightly discourages the use of forward references, which deep in my heart I know is right.

Enough about forward references, let's talk for a moment about immediate words. Immediate words present a special problem since they must be executed at compile time. They may do arbitrarily crazy things, and must do them in the target environment. For example, ['] must look up the next word in the input stream and compile its code field as literal. Another example is ." which must scan the input until another " is encountered, and then compile the runtime address for (." which may not even be known yet, followed by the count-delimited string that was scanned. The usual mechanism used to implement immediate words is through a new defining word called τ : which behaves just like Forth's : except that the definition it creates is placed in the target vocabulary, or symbol table. As you recall from last time, the main compiling loop looks up words in the symbol table and executes them. Words that are defined by **CODE** and : are placed in this symbol table, and when executed compile themselves. By using τ : we can place words into this symbol table that do things other than compile themselves. For example ." would have to first compile the run time for ." namely (.", and then get the string and compile it into the target image. This is totally different behaviour from, say, the meta version of **DUP**, which simply compiles a pointer to the code field of **DUP** when it is executed. Thus, for each

immediate word that passes through the meta compiling process, we must define a special case compiling word that "does the right thing" in the meta context.

Now I must apologize for not providing any code this time around. The problem is that all of the issues I discussed above are implemented in a very system-dependent manner; hence I would have to make a lot of assumptions about exactly how vocabularies work and how different system details operate. Rather than do that and provide code that would not run on any existing systems, I decided not to provide any code, but simply to discuss some of the remaining concepts involved in meta compiling. The best way to really learn about meta compilers is to write one. Hopefully, I have provided you with enough ammunition to attempt such an undertaking. Let me tell you that if you do, you will raise your level of Forth consciousness many

levels, and I think it is an exercise well worth the effort.

Next time I will talk about multi-tasking, an issue many have heard about but few have seen. We will implement a very simple (and slow) high-level multi-tasker and discover its principles of operation. Until then, good luck and may Forth be with you!

Copyright ©1983 by Henry Laxen. All rights reserved. The author is Chief Software Engineer for Universal Research, 150 North Hill Drive #10, Brisbane, CA 94005, specializing in the development of portable computers.

FORTH Vendors (Continued from page 39)

InnoSys
2150 Shattuck Ave.
Berkeley, CA 94704
415/843-8114

Consultation & Training Only
See System Vendor Chart
for others

Bartholomew, Alan
2210 Wilshire Blvd. #289
Santa Monica, CA 90403
213/394-0796

Boulton, Dave
581 Oakridge Dr.
Redwood City, CA 94062

Brodie, Leo
9720 Baden Ave.
Chatsworth, CA 91311
213/998-8302

Eastgate Systems Inc.
P.O. Box 1307
Cambridge, MA 02238

Girton, George
1753 Franklin
Santa Monica, CA 90404
213/829-1074

Go FORTH
504 Lakemead Way
Redwood City, CA 94062
415/366-6124

Harris, Kim R.
Forthright Enterprises
P.O. Box 50911
Palo Alto, CA 94303
415/858-0933

Intersystems Management
Computer Consultancy
Story Hill Rd. RFD3
Dunbarton, NH 03045
603/774-7762

Laxen, Henry H.
1259 Cornell Ave.
Berkeley, CA 94706
415/525-8582

McIntosh, Norman
2908 California Ave., #3
San Francisco, CA 94115
415/563-1246

Metalogic Corp.
4325 Miraleste Dr.
Rancho Palos Verdes, CA 90274
213/519-7013

Peschke, Manfred
Intersystems Mgmt. & Consult.
Story Hill Rd. RFD 3
Dunbarton NH 03045
603/774-7762

Petri, Martin B.
Computer Consultants
16005 Sherman Way
Suite 104
Van Nuys, CA 91406
213/908-0160

Redding Co.
P.O. Box 498
Georgetown, CT 06829
203/938-9381

Schleisiek, Klaus
Eppendorfer Landstr. 16
D 2000 Hamburg 20
West Germany
(040)480 8154

Schrenk, Dr. Walter
Postfach 904
7500 Karlstruhe-41
West Germany

Software Engineering
6308 Troost Ave. #210
Kansas City, MO 64131
816/363-1024

Softweaver
P.O. Box 7200
Santa Cruz, CA 95061
408/425-8700

Technology Management, Inc.
1520 S. Lyon St.
Santa Ana, CA 92705
714/835-9512

Timin, Mitchel
3050 Rue d'Orlean #307
San Diego, CA 92110
619/222-4185

CompuPro's System 8/16

Because Computers Shouldn't

...slow down. CompuPro's System 8/16 doesn't nap on the bus. And it's not just fast. It's fast and CP/M versatility together. It's got 256K of RAM, 2.4 megabytes of disk storage, and a whole lot of ultra-fast RAM-based disk memory. You can even plug into the four serial ports, one CompuPro's got it all. Scientists, industrial integrators and software engineers choose CompuPro for performance, quality and reliability. Also available in 8000 and 16000 configurations.

RAM 22

Because Random Access Memory Shouldn't Bloat.

When your random access memory is acting sluggish and slow, CompuPro's RAM 22 provides a whole lot of extra space. You get an extra 256K x 8 of random access memory, on a state-of-the-art board that's clocked at 12 MHz. RAM 22 draws half the power of other equivalents, but offers the speed and reliability of fully static operation. \$2,495 (ASM), \$2,995 (CSC).

CPU 88K

Because Processing Speeds Shouldn't Bland.

Ever have the feeling your computer isn't in the hands of a blindingly brilliant CompuPro's indomitable CPU 88K. It's got the efficiency to your programming tasks, it operates up to 10 MHz, and works with both 8- and 16-bit memory. \$695 (ASM), \$950 (CSC).



CompuPro products are compatible with all S-100/IEEE 696 hardware. For performance, quality and reliability, contact your nearest Full Service CompuPro System Center today; call (415) 786-0909 extension 206 for location.



CompuPro, A **GODDOUT** COMPANY
3506 Breakwater Ct., Hayward, CA 94545

FIG Chapter News

John D. Hall
Oakland, California

We now have forty-four chapters! The six new chapters are:

Rockwell FIG Chapter
Hoffman Estates, Illinois

Wichita FIG Chapter (FIGPAC)
Wichita, Kansas

Kansas City FIG Chapter
Kansas City, Missouri

Colombia FIG Chapter
Bogota, Colombia

Forth Interest Group — U.K.
London, England

Taiwan FIG Chapter
Taipei, Taiwan

Orange County Chapter

At the June 22 meeting, Dr. David E. Winkel of the University of Wyoming drew a full house. He spoke primarily about computer processor development and bit-slice design. He also introduced a bit-slice development system which he developed. There was a short discussion about an HP-75 Forth which one member is developing.

Wil Baden went through the fundamentals of conditionals at the July 6 meeting. Near the end of the meeting, Wil introduced non-compiling conditionals which allow one to occasionally compile. Also, Greg Stevenson developed a compiling buffer approach for conditionals. Bob Waters demonstrated Forth on a Timex-Sinclair, as advertised in Forth Dimensions. The Forth was direct compiling, multi-tasking, and had windowing capability.

On July 27, Wil Baden presented the **ONLY ALSO** vocabulary concepts of William F. Ragsdale. Many were already using the concept and were pleased with its performance. Wil added some

財團 資訊工業策進會贊助
法人

FORTH

系統研習會

為發展最新電腦資訊技術，推廣最具威力 FORTH 語言。本會特邀請旅美學人，美國洛克希德飛機公司非破壞性檢驗主任丁陳漢蓀博士專程返國（七月二十日抵台，二十六日返美）主講最新電腦專題，機會難得，歡迎儘早報名參加，俾便準備講義及安排講習日期內午餐。

時間：七月廿一日至七月二十五日

每天 上午 9:00 ~ 12:00 (專題講座)

下午 1:30 ~ 3:30 (研討會)

報名地點：東方電腦中心(台北市重慶南路一段 121 號東方出版社三樓)

電話：3311316, 3814907 ~ 08

研討會地址：資訊工業策進會研討室

台北市南京東路二段 116 號亞信大樓 12 樓

專題：● Poly-FORTH 及 1983 年 FORTH 新標準介紹

● Multitasking & multiprogramming

● Target compilation

● Case study (Control)

● Case study (Database)

費用：每人新台幣伍仟元 (包括午餐及講義)

主辦 世界 FORTH 學會(FIG)台灣分會
(中華民國 FORTH 語言學會)

協辦 福世資訊科技公司
東方電腦中心

台北市重慶南路 1 段 121 號 3 樓
(衡陽路口東方出版社三樓)

News from the FIG Chapter of Taiwan

words that increased its performance. Zane Thomas is implementing a moderm system for the Orange County Chapter to transfer screens. Bob Snook presented a short discussion on alternatives to **CASE** statements.

William Vock, who was visiting Greg Stevenson and collaborating with him on the development of software for the Epson QX-10 computer, presented a graphics package on August 3. Its performance was impressive. Mr. Vock is a graphics expert and knows the subject very well: he had done it with 8K of Forth.

Rochester Chapter

The Rochester (New York) FIG Chapter had its second meeting on June 25 at the University of Rochester. The group looked at the new 83-Standard draft, and Larry Forsley repeated the talk Bob Smith gave at the Rochester Conference on the specific differences between Forth-79 and 83-Standard. There was much discussion of the implications for ROMmed systems of the new mono-addressing rules. Specifical, does **BODY**> reference an address in RAM or ROM? It was suggested that copies of Bob's slides accompany distributed copies of the new standard.

Nova Scotia Chapter

The Nova Scotia Chapter held their first meeting on June 29. The group took a survey at this meeting and found (as in most chapters) that the members had a wide range of experience with Forth. Some people had no Forth experience at all, while others had written meta-compilers. They decided, for purposes of code exchange, that they would use 79-Standard. People were "volunteered" (a common Forth practice) to look into transferring screens to dissimilar computers and into collecting a list of all Forth information owned by various members in order to generate a master list. Graphics standards were also discussed and, as with any standards discussion, grew heated. As the bell rang, each side went to their respective corners. We will hear more about this in the future!

Dayton Chapter

At the June 25 meeting, Dr. Leonard Spialter gave a slide show of the Rochester conference. On July 12, the group decided to have a Forth tutorial and Dr. Spialter presented a flow chart for a "Day of the Week" program. The group spent the rest of the meeting programming it in Forth. Gary Granger gave a talk about Forth to the Columbus Ohio Heath Users Group on July 11, and a Forth talk and demo to the Timex-Sinclair Users group on July 19.

Kansas City Chapter

The first two meetings of the Kansas City Chapter had twenty to twenty-five

attendees. The first meeting was spent getting acquainted and discovering who was doing what with Forth. At the second meeting, the group generated a list of topics of special interest for groups or programs. The highest items on the list were: program organization/coding style, graphics, data-base applications, and target compilers.

Support your local chapter!

John D. Hall is the Chapter Coordinator for the Forth Interest Group and is a consulting programmer.

Chapters in Formation

Here are more of the new chapters that are forming. If you live in any of these areas, contact one of these people and offer your support in forming a FIG chapter.

Contact:

Michael Perry
1446 Stannage Ave.
Berkeley, CA 94702

Dick Turpin
3109 Breton Ave.
Davis, CA 95616

Samuel J. Cook
115 N. Washington Ave.
Batavia, IL 60510

Dr. Edward Newberger
2739 Elmwood Ave., Apt. 3
Kenmore, NY 14217

David Whitely
1163 West 550 North
Clearfield, UT 84015

Arnold Pinchuk
2130 Menasha Ave.
Manitowoc, WI 54220

T. William Rudolph
FIG-GRAPH East
592 Plymouth St.
Halifax, MA 02338

Tony Van Muyden
P.O. Box 7396
Edmonton, Alberta
T5E 6C8 Canada

Jack Hung
Comx World Operations
15/F Wo Kee Hong Bldg.
585-609 Castle Peak Rd.
Kwai Chung, N.T.
Hong Kong

Ravizza Donato
Sonnenbergstr. 34A
Uster 8610
Switzerland

Greg Stevenson
8002 Poinsettia Place
Buena Park, CA 90620

Glen Bowie
25746 North Player Dr., #Q-1
Valencia, CA 91355

Marc Perkel
Perkel Software Systems
1452 N. Clay
Springfield, MO 65802
(417) 862-9830

H. Marcus Bacon
704-1H E.I. DuPont
Savannah River Plant
Aiken, SC 29808

Richard Bloch
Eastern VA Center for MH Studies
Drawer A
Williamsburg, VA 23187

Wes Thomas
Jupiter Ace SIG
Frank Barth, Inc.
500-5th Ave.
New York, NY 10110

Scott Miles
Robotics
Christensen Diamond Products
2532 South 3270 West
Salt Lake City, UT 84119

Erick Ostergaard
COMPEX
2 Gertsvej
2300 Copenhagen S.
Denmark

Marc (Tamir) Weiner
Moshav Neve Ilan
D.N. Harei Yehuda 90850
Israel

New Product Announcements

Forth Dimensions welcomes press releases and product announcements, as well as reader letters regarding product performance. Addresses of the distributors and manufacturers mentioned in this column may be found in the Vendors List section.

The latest Forth news from Little Rock is that Hawg Wild Software offers the **XFORTH XCHANGE** to original users of the XFORTH Forth-79 product. Questions, ideas and implementations should be sent to that company, who says their newest service will be "free and unrestricted."

Atari owners will be interested in **Power Forth** (for the 800/800XL, and 1200XL) from Elcomp Publishing. It is an extended FIG-Forth with editor and I/O routines. The utilities package includes decompiler, sector copy, Atari filehandling, graphics and sound, joystick program and player missile. The \$39.95 price also covers two game demos and a mailing list application. Floating point with trig is an added \$29.95, and the beginners' subset "Learn Forth" (requires 32K for disk or 16K for tape version) costs \$19.95.

"UNIX-like word processing in Forth" is the claim made for **Forth-ms**. The licensed source code runs on Apple

II computers using Epson printers with Graphtrax Plus, but can be configured by the user for other printers. Print spooling allows "simultaneous" use of printer and keyboard, and Greek letters and other symbols are available by typing a command followed by the name of the desired letter or symbol. The price (in single quantity) is \$200 from Innovatia Laboratories.

The TDS900 is a **single-board Forth computer** with on-board screen editor, compiler and debug facilities. It uses FIG-Forth and provides simple interface to serial and parallel devices. All the user needs is a power supply, CRT and \$395. If more than 12K RAM and 8K ROM is needed, up to 160K is available in increments of 20K per extra board. The computer and RAM boards use CMOS throughout, in single-Eurocard format. Information is available from Stynetic Systems, Inc. in the U.S. and from Triangle Digital Services, Ltd. in the U.K.

Forth classes will demonstrate how Forth can be used as an algorithm development tool and as a total programming environment. Problem solving will be emphasized by instructor Leo Brodie, author of *Starting Forth*.

Students will apply design and problem-solving techniques in the design and coding of actual problems. East coast classes are planned for November, and the Los Angeles area will be covered in January.

"Any desired data file format" can benefit from **INDEX+**. By using its B-Tree ISAM utilities, Forth programmers can create and maintain keyed indexes in order to perform searches randomly, or sequentially in either direction. The program supports **BLOCK** disk I/O and the CP/M and MS-DOS interface by Laboratory Microsystems. Retail orders (the price is \$125) should be sent to Laboratory Microsystems, for whose Forth systems **INDEX+** is written; others should contact Business Computing Press.

Sylmar Software now offers **FIG-Forth** for the **Otrona Attache**. The two-disk set costs \$50 and includes a full-screen editor and various utilities. A Towers of Hanoi version demonstrates the Attache's direct cursor operations. The user should obtain the FIG-Forth Installation Manual, which provides definitions for the Forth words.

Fig Chapters

U.S.

• ARIZONA

Phoenix Chapter
Call Dennis L. Wilson
602/956-7678

• CALIFORNIA

Los Angeles Chapter
Monthly, 4th Sat., 11 a.m.
Allstate Savings
8800 So. Sepulveda Boulevard
Los Angeles
Call Phillip Wasson
213/649-1428

Northern California Chapter
Monthly, 4th Sat., 1 p.m.
FORML Workshop at 10 a.m.

Palo Alto area.
Contact FIG Hotline
415/962-8653

Orange County Chapter
Monthly, 4th Wed., 7 p.m.
Fullerton Savings
Talbert & Brookhurst
Fountain Valley
Monthly, 1st Wed., 7 p.m.
Mercury Savings
Beach Blvd. & Eddington
Huntington Beach
Call Noshir Jesung
714/842-3032

San Diego Chapter
Weekly, Thurs., 12 noon.
Call Guy Kelly
619/268-3100 ext. 4784

• COLORADO

Denver Chapter
Monthly, 1st Mon., 7 p.m.
Call Steven Sarns
303/477-5955

• ILLINOIS

Rockwell Chicago Chapter
Call Gerard Kusiolek
312/885-8092

• KANSAS

Wichita Chapter (FIGPAC)
Monthly, 3rd Wed., 7 p.m.
Wilber E. Walker Co.
532 S. Market
Wichita, KS
Call Arne Flones
316/267-8852

• MASSACHUSETTS

Boston Chapter
Monthly, 1st Wed., 5 p.m.
Mitre Corp. Cafeteria
Bedford, MA
Call Bob Demrow
617/688-5661 after 7 p.m.

• MICHIGAN

Detroit Chapter
Call Dean Vieau
313/493-5105

• MINNESOTA

MNFIG Chapter
Monthly, 1st Mon.
1156 Lincoln Avenue
St. Paul, MN
Call Fred Olson
612/588-9532

• **MISSOURI**

Kansas City Chapter
Call Terry Rayburn
816/363-1024

St. Louis Chapter
Monthly, 3rd Tue., 7 p.m.
Thornhill Branch of
St. Louis County Library
Call David Doua
314/867-4482

• **NEVADA**

Southern Nevada Chapter
Suite 900
101 Convention Center Drive
Las Vegas, NV
Call Gerald Hasty
702/453-3544

• **NEW JERSEY**

New Jersey Chapter
Call George Lyons
201/451-2905 eves.

• **NEW YORK**

New York Chapter
Monthly, 2nd Wed., 8 p.m.
Queens College
Call Tom Jung
212/432-1414 ext. 157 days
212/261-3213 eves.

Rochester Chapter
Monthly, 4th Sat., 2 p.m.
Hutchison Hall
Univ. of Rochester
Call Thea Martin
716/235-0168

Syracuse Chapter
Call C. Richard Corner
315/456-7436

• **OHIO**

Athens Chapter
Call Isreal Urieli
614/594-3731

Dayton Chapter
Twice monthly, 2nd Tues &
4th Wed., 6:30 p.m.
CFC, 11 W. Monument Ave.
Suite 612
Dayton, OH
Call Gary M. Granger
513/849-1483

• **OKLAHOMA**

Tulsa Chapter
Monthly, 3rd Tues., 7:30 p.m.
The Computer Store
4343 South Peoria
Tulsa, OK
Call Art Gorski
918/743-0113

• **OREGON**

Greater Oregon Chapter
Monthly, 2nd Sat., 1 p.m.
Computer & Things
3460 SW 185th, Aloha
Call Timothy Huang
503/289-9135

• **TEXAS**

**Dallas/Ft. Worth
Metroplex Chapter**
Monthly, 4th Thurs., 7 p.m.
Software Automation, Inc.
14333 Porton, Dallas
Call Marvin Elder
214/392-2802 or
Bill Drissel
214/264-9680

San Antonio Chapter
T.L. Schneider
8546 Broadway, Suite 207
San Antonio, TX 78217

• **VERMONT**

Vermont Fig Chapter
Monthly, 4th Thurs., 7:30 p.m.
The Isley Library, 3rd fl.
3rd Floor Meeting Room
Middleburynes, VT
Call Hal Clark
802/877-2911 days
802/452-4442 eves

• **VIRGINIA**

Potomac Chapter
Monthly, 1st Tues., 7 p.m.
Lee Center
Lee Highway at Lexington St.
Arlington, VA
Call Joel Shprentz
703/437-9218 eves.

FOREIGN

• **AUSTRALIA**

Australia Fig Chapter
Contact: Ritchie Laird
25 Gibsons Road
Sale, Victoria 3850
051/44-3445

FIG Australia Chapter
Contact: Lance Collins
65 Martin Road
Glen Iris, Victoria 3146
03/29-2600

Sydney Chapter
Monthly, 2nd Fri., 7 p.m.
Morven Brown Bldg., Rm LG16
Univ. of New South Wales
Sydney
Contact: Peter Tregaele
10 Binda Rd., Yowie Bay
02/524-7490

• **BELGIUM**

Belgium Chapter
Contact: Luk Van Loock
Lariksdreff 20
B2120 Schoten
03/658-6343

• **CANADA**

Nova Scotia Chapter
Contact: Howard Harawitz
P.O. Box 688
Wolfville, Nova Scotia B0P 1X0
902/542-7812

Southern Ontario Chapter
Monthly, 1st Sat., 2 p.m.
General Sciences Bldg, Rm 312
McMaster University
Contact: Dr. N. Solntseff
Unit for Computer Science
McMaster University
Hamilton, Ontario L8S 4K1
416/525-9140 ext. 2065

Quebec Chapter
Call Gilles Paillard
418/871-1960 or
418/643-2561

• **COLOMBIA**

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

• **ENGLAND**

Forth Interest Group -- U.K.
Monthly, 1st Thurs.,
7 p.m., Rm. 408
Polytechnic of South Bank
Borough Rd., London
Contact: Keith Goldie-Morrison
15 St. Albans Mansion
Kensington Court Place
London W8 5QH

• **ITALY**

FIG Italia
Contact: Marco Tausel
Via Gerolamo Forni 48
20161 Milano
02/645-8688

• **NETHERLANDS**

**HCC-FORTH Interest
Group Chapter**
F.J. Meijer
Digicos
Aart V.D. Neerweg 31
Ouderkerk A.D.
Amstel, The Netherlands

• **SOUTH AFRICA**

Contact: Edward Murray
Forthwith Computers
P.O. Box 27175
Sunnyside, Pretoria 0132

• **SWITZERLAND**

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

• **TAIWAN**

Taiwan Chapter
Contact: J.N. Tsou
Forth Information Technology
P.O. Box 53-200
Taipei
02/331-1316

• **WEST GERMANY**

West German Chapter
Klaus Schleisiek
FIG Deutschland
Postfach 202264
D 2000 Hamburg 20
West Germany

SPECIAL GROUPS

**Apple Corps FORTH
Users Chapter**
Twice Monthly, 1st &
3rd Tues., 7:30 pm
1515 Sloat Boulevard, #2
San Francisco, CA
Call Robert Dudley Ackerman
415/626-6295

Baton Rouge Atari Chapter
Call Chris Zielewski
504/292-1910

FIGGRAPH
Call Howard Pearlmutter
408/425-8700

MMSFORTH Users Groups
Monthly, 3rd Wed., 7 p.m.
Cochituate, MA
Dick Miller
617/653-6136
(25 groups world-wide)

FORTH System Vendors

(by Category)

(Codes refer to alphabetical listing e.g., A1 signifies AB Computers, etc.)

Processors

1802	C2, C3, F3, F6, L3
6502 (AIM, KIM, SYM)	R1, R2, S2
6800	C3, F3, F5, K1, L3, M6, T1
6801	P4
6809	C3, F3, L3, M6, T1
68000	C3, C5, D1, E1, K1
68008	P4
8080/85	A5, C2, C3, F4, I5, L1, L3, M3, M6, R1, T3
Z80/89	A3, A5, C3, F4, I3, L1, M2, M3, M5, N1, T3
Z80000	I3
8086/88	C3, F2, F3, L1, L3, M6
9900	E4, L3

Operating Systems

CP/M	A3, A5, C3, F3, I3, L3, M1, M2, M6, T3
CP/M-86	C3

Computers

Alpha Micro	P3, S4
Apple	A4, E1, E2, F4, I2, I5, J1, L4, M2, M6, M8, O2, O3 E1, E2, M6, P2, Q1, V1
Atari	
Compaq	M5
Cromemco	A5, M2, M6
DEC PDP/LSI-11	C3, F3, L2, S4
Heath-89	M2, M6, M7
Hewlett-Packard 85	
Hewlett-Packard 9826/36	C5
IBM PC	A8, C3, F3, L1, M5, M6, Q2, S8, W2
IBM Other	L3, W1
Kaypro II/Xerox 820	M2
Micropolis	A2, M2, S3
North Star	I5, M2, P1, S11
Nova	C6
Ohio Scientific	A6, B1, C4, O1, S7, T2
Osborne	M2
Pet SWTPC	A1, A6, B1, C3, O1, S7, T2, T5
Poly Morphic Systems	A7
TRS-80 I, II, and/or III	I5, M2, M5, M6, S5, S6, S9
TRS-80 Color	A3, A8, F5, M4, T1
Vector Graphics	M2

Other Products/Services

Applications	P4
Boards, Machine	F3, M3, P4, R2, S10
Consultation	C3, C5, N1, P4, T3, W1
Cross Compilers	C3, F3, I3, M6, N1, P4
Products, Various	A5, B2, C3, C7, F3, I4, I5, S8, S12, W2
Training	C3, F3, I3, P4, W1

FORTH Vendors (Alphabetical)

The following vendors offer FORTH systems, applications, or consultation. FIG makes no judgment on any product, and takes no responsibility for the accuracy of this list. We encourage readers to

keep us informed on availability of the products and services listed. Vendors may send additions and corrections to the Editor, and must include a copy of sales literature or advertising.

FORTH Systems

A	B
1. AB Computers 252 Bethlehem Pike Colmar, PA 18915 215/822-7727	1. Blue Sky Products 729 E. Willow Signal Hill, CA 90806
2. Acropolis 17453 Via Valencia San Lorenzo, CA 94580 415/276-6050	2. Business Computing Press 2210 Wilshire Blvd. Suite 289 Santa Monica, CA 90403 213/394-0796
4. Applied Analytics Inc. 8910 Brookridge Dr., #300 Upper Marlboro, MD 20870	C
5. Aristotelian Logicians 2631 E. Pinchot Ave. Phoenix, AZ 85016	1. Capstone Computing, Inc. 5640 Southwyck Blvd., #2E Toledo, OH 43614 419/866-5503
7. Abstract Systems, etc. RFD Lower Prospect Hill Chester, MA 01011	2. Chrapkiewicz, Thomas 16175 Stricker East Detroit, MI 48021
8. Armadillo Int'l Software P.O. Box 7661 Austin, TX 78712 512/459-7325	3. CMOSoft P.O. Box 44037 Sylmar, CA 91342

4. COMSOL, Ltd. Treyway House Hanworth Lane Chertsey, Surrey England KT16 9LA	2. Elcomp-Hofacker Tegernseerstr. 18 D-8150 Holzkirchen West Germany 08024/7331 Telex 52 69 73
5. Consumer Computers 8907 La Mesa Blvd. La Mesa, CA 92041 714/698-8088	3. Emperical Research Group P.O. Box 1176 Milton, WA 98354 206/631-4855
6. Creative Solutions, Inc. 4801 Randolph Rd. Rockville, MD 20852 301/984-0262	4. Engineering Logic 1252 13th Ave. Sacramento, CA 95822
7. Curry Associates P.O. Box 60324 Palo Alto, CA 94306	

E

1. Elcomp Publishing, Inc. 53 Redrock Lane Pomona, CA 91766 (714) 623-8314 Telex 29 81 91

F

1. Fantasia Systems, Inc. 1059 The Alameda Belmont, CA 94002 415/593-5700	3. FORTH, Inc. 2309 Pacific Coast Highway Hermosa Beach, CA 90254 213/372-8493
--	---

4. FORTHWare
639 Crossridge Terrace
Orinda, CA 94563
5. Frank Hogg Laboratory
130 Midtown Plaza
Syracuse, NY 13210
315/474-7856
6. FSS
P.O. Box 8403
Austin, TX 78712
512/477-2207
- H**
1. HAWG WILD Software
P.O. Box 7668
Little Rock, AR 72217
- I**
1. IDPC Company
P.O. Box 11594
Philadelphia, PA 19116
215/676-3235
2. IUS (Cap'n Software)
281 Arlington Ave.
Berkeley, CA 94704
415/525-9452
3. Inner Access
517K Marine View
Belmont, CA 94002
415/591-8295
4. Innovatia Laboratories
5275 Crown St.
West Linn, OR 97068
5. Insoft
10175 S.W. Barbur Blvd.
Suite #202B
Portland, OR 97219
503/244-4181
6. Interactive Computer
Systems, Inc.
6403 Di Marco Rd.
Tampa, FL 33614
- J**
1. JPS Microsystems, Inc.
361 Steelcase Rd., W.
Markham, Ontario
Canada L3R 3V8
416/475-2383
- K**
1. Kukulies, Christoph
Ing. Buro Datentec
Heinrichsallee 35
Aachen, 5100
West Germany
- L**
1. Laboratory Microsystems
4147 Beethoven St.
Los Angeles, CA 90066
213/306-7412
2. Laboratory Software
Systems, Inc.
3634 Mandeville Canyon
Los Angeles, CA 90049
213/472-6995
3. Lynx
3301 Ocean Park, #301
Santa Monica, CA 90405
213/450-2466
4. Lyons, George
280 Henderson St.
Jersey City, NJ 07302
201/451-2905
- M**
1. M & B Design
820 Sweetbay Dr.
Sunnyvale, CA 94086
2. MicroMotion
12077 Wilshire Blvd., #506
Los Angeles, CA 90025
213/821-4340
3. Microsystems, Inc.
2500 E. Foothill Blvd., #102
Pasadena, CA 91107
213/577-1477
4. Micro Works, The
P.O. Box 1110
Del Mar, CA 92014
714/942-2400
5. Miller Microcomputer
61 Lake Shore Rd.
Natick, MA 01760
617/653-6136
6. Mountain View Press
P.O. Box 4656
Mountain View, CA 94040
415/961-4103
7. MCA
8 Newfield Ln.
Newtown, CT 06470
8. Metacrafts Ltd.
Beech Trees, 144 Crewe Rd.
Shavington, Crewe
England CW1 5AJ
- N**
1. Nautilus Systems
P.O. Box 1098
Santa Cruz, CA 95061
408/475-7461
- O**
1. OSI Software & Hardware
3336 Avondale Court
Windsor, Ontario
Canada N9E 1X6
519/969-2500
2. Offete Enterprises
1306 S "B" St.
San Mateo, CA 94402
3. On-Going Ideas
RD #1, Box 810
Starksboro, VT 05487
802/453-4442
- P**
1. Perkel Software Systems
1636 N. Sherman
Springfield, MO 65803
2. Pink Noise Studios
P.O. Box 785
Crockett, CA 94525
415/787-1534
3. Professional Mgmt. Services
724 Arastradero Rd., #109
Palo Alto, CA 94306
408/252-2218
4. Peopleware Systems Inc.
5190 West 76th St.
Minneapolis, MN 55435
612/831-0827
- Q**
1. Quality Software
6660 Reseda Blvd., #105
Reseda, CA 91335
2. Quest Research, Inc.
P.O. Box 2553
Huntsville, AL 35804
800/558-8088
- R**
2. Rockwell International
Microelectronics Devices
P.O. Box 3669
Anaheim, CA 92803
714/632-2862
- S**
1. Satellite Software Systems
288 West Center
Orem, UT 84057
801/224-8554
2. Saturn Software, Ltd.
P.O. Box 397
New Westminster, BC
Canada V3L 4Y7
3. Shaw Labs, Ltd.
P.O. Box 3471
Hayward, CA 94540
415/276-6050
4. Sierra Computer Co.
617 Mark NE
Albuquerque, NM 87123
5. Sirius Systems
7528 Oak Ridge Highway
Knoxville, TN 37921
615/693-6583
6. Software Federation
44 University Drive
Arlington Hts., IL 60004
312/259-1355
7. Software Works, The
1032 Elwell Ct., #210
Palo Alto, CA 94303
415/960-1800
8. Spectrum Data Systems
5667 Phelps Luck Dr.
Columbia, MD 21045
301/992-5635
9. Stearns, Hoyt Electronics
4131 E. Cannon Dr.
Phoenix, AZ 85028
602/996-1717
10. Stynetic Systems, Inc.
Flowerfield, Bldg. 1
St. James, NY 11780
516/862-7670
11. Supersoft Associates
P.O. Box 1628
Champaign, IL 61820
217/359-2112
12. Sylmar Software
P.O. Box 44037
Sylmar, CA 91342
- T**
1. Talbot Microsystems
1927 Curtis Ave.
Redondo Beach, CA 90278
2. Technical Products Co.
P.O. Box 12983
Gainesville, FL 32604
904/372-8439
3. Timin Engineering Co.
C/o Martian Technologies
8348 Center Dr. Suite F
La Mesa, CA 92041
619/464-2924
4. Transportable Software
P.O. Box 1049
Hightstown, NJ 08520
609/448-4175
- V**
1. Valpar International
3801 E. 34th St.
Tucson, AZ 85713
800/528-7070
- W**
1. Ward Systems Group
8013 Meadowview Dr.
Frederick, MD 21701
2. Worldwide Software
2555 Buena Vista Ave.
Berkeley, CA 94708
415/644-2850
- Z**
1. Zimmer, Tom
292 Falcato Dr.
Milpitas, CA 95035
- Boards & Machines Only**
See System Vendor Chart
for others
- Controlex Corp.
16005 Sherman Way
Van Nuys, CA 91406
213/780-8877
- Datricon
7911 NE 33rd Dr., #200
Portland, OR 97211
503/284-8277
- Golden River Corp.
7315 Reddfield Ct.
Falls Church, CA 22043
- Triangle Digital Services Ltd.
23 Campus Road
London E17 5PG
England
- Application Packages Only**
See System Vendor Chart
for others
- Curry Associates
P.O. Box 11324
Palo Alto, CA 94306
415/322-1463

(Continued on page 32)

FORTH INTEREST GROUP

MAIL ORDER

	USA	FOREIGN AIR
<input type="checkbox"/> Membership in FORTH Interest Group and Volume V of FORTH DIMENSIONS	\$15	\$27
<input type="checkbox"/> Back Volumes of FORTH DIMENSIONS. Price per each.	\$15	\$18
<input type="checkbox"/> I <input type="checkbox"/> II <input type="checkbox"/> III <input type="checkbox"/> IV		
<input type="checkbox"/> fig-FORTH Installation Manual, containing the language model of fig-FORTH, a complete glossary, memory map and installation instructions	\$15	\$18
<input type="checkbox"/> Assembly Language Source Listings of fig-FORTH for specific CPUs and machines. The above manual is required for installation.	\$15	\$18
Check appropriate box(es). Price per each.		
<input type="checkbox"/> 1802 <input type="checkbox"/> 6502 <input type="checkbox"/> 6800 <input type="checkbox"/> 6809 <input type="checkbox"/> VAX <input type="checkbox"/> z80		
<input type="checkbox"/> 8080 <input type="checkbox"/> 8086/8088 <input type="checkbox"/> 9900 <input type="checkbox"/> APPLE II <input type="checkbox"/> ECLIPSE		
<input type="checkbox"/> PACE <input type="checkbox"/> NOVA <input type="checkbox"/> PDP-11 <input type="checkbox"/> 68000 <input type="checkbox"/> ALPHA MICRO		
<input type="checkbox"/> "Starting FORTH, by Brodie. BEST book on FORTH. (Paperback)	\$18	\$22
<input type="checkbox"/> "Starting FORTH" by Brodie. (Hard Cover)	\$23	\$28
<input type="checkbox"/> PROCEEDINGS: FORML (FORTH Modification Conference)		
<input type="checkbox"/> 1980, \$25USA/\$35Foreign		
<input type="checkbox"/> 1981, Two Vol., \$40USA/\$55Foreign		
<input type="checkbox"/> 1982, \$25USA/\$35Foreign		
ROCHESTER FORTH Conference		
<input type="checkbox"/> 1981, \$25USA/\$35Foreign		
<input type="checkbox"/> 1982, \$25USA/\$35Foreign		
<input type="checkbox"/> 1983, \$25USA/\$35Foreign	Total	\$ _____
<input type="checkbox"/> STANDARD: <input type="checkbox"/> FORTH-79, <input type="checkbox"/> FORTH-83. \$15USA/\$18Foreign EACH.	Total	\$ _____
<input type="checkbox"/> Kitt Peak Primer, by Stevens. An in-depth self-study book.	\$25	\$35
<input type="checkbox"/> MAGAZINES ABOUT FORTH: <input type="checkbox"/> BYTE Reprints 8/80-4/81		
<input type="checkbox"/> Dr Dobb's Jrnl, <input type="checkbox"/> 9/81, <input type="checkbox"/> 9/82, <input type="checkbox"/> 9/83		
<input type="checkbox"/> Poplar Computing, 9/83 \$3.50USA/\$5Foreign EACH.	Total	\$ _____
<input type="checkbox"/> FIG T-shirts: <input type="checkbox"/> Small <input type="checkbox"/> Medium <input type="checkbox"/> Large <input type="checkbox"/> X-Large	\$10	\$12
<input type="checkbox"/> Poster, BYTE Cover 8/80, 16"x22"	\$ 3	\$ 5
<input type="checkbox"/> FORTH Programmer's Reference Card. If ordered separately, send a stamped, self addressed envelope.		Free
TOTAL	\$ _____	

NAME _____ MS/APT _____

ORGANIZATION _____ PHONE() _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____ COUNTRY _____

VISA# _____ MASTERCARD# _____

AMERICAN EXPRESS# _____ Card Expiration Date _____

(Minimum of \$15.00 on Charge Cards)

Make check or money order in US Funds on US Bank, payable to: FIG. All prices include postage. No purchase orders without check. California residents add sales tax. 10/83

ORDER PHONE NUMBER: (415) 962-8653

FORTH INTEREST GROUP * PO BOX 1105 * SAN CARLOS, CA 94070

FORTH INTEREST GROUP

P.O. Box 1105
San Carlos, CA 94070

BULK RATE U.S. POSTAGE PAID Permit No. 261 Mt. View, CA

ROBERT SMITH
2300 ST. FRANCIS DR
PALO ALTO, CA 94303