

# F O R T H

---

D I M E N S I O N S

■  
*HEADLESS COMPILER*

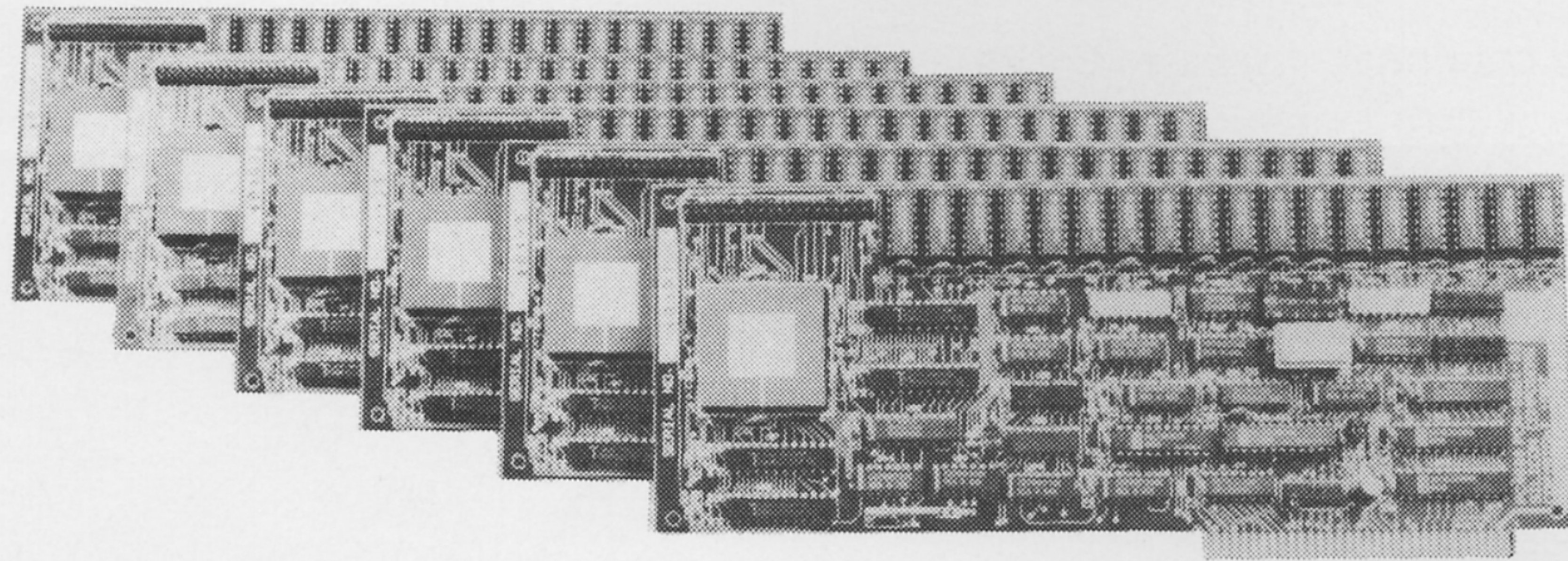
*STARTING FORTH INC.*

*THIRTY-TWO-BIT FORTH*

*RACTAL LANDSCAPES*  
■

# **INTRODUCING THE PC-RISC SYSTEM**

**For High-Speed Concurrent Parallel Processing  
ADD UP TO 30 MIPS TO YOUR PC!**



## **THE PC-RISC SYSTEM**

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

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

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

Formerly  
**SOFTWARE COMPOSERS**



---

**SILICON COMPOSERS**

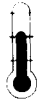
# F O R T H

D I M E N S I O N S

---

## *FRACTAL LANDSCAPES • BY PHIL KOOPMAN, JR.*

12



This program generates fractal landscapes, with height-based coloring and hidden-surface elimination. Fractals are used to describe various geometrical shapes, especially natural phenomena, that are not strictly one, two, or three-dimensional.

---

## *FORTH TO THE FUTURE • BY MITCH BRADLEY*

17



Our new columnist discusses the growing importance of 32-bit machines, and presents a proven scheme for implementing Forth on them. This method allows programs to run unchanged on either 16-bit or 32-bit machines without penalizing the newer architectures.

---

## *STARTING FORTH INC. • INTERVIEW WITH ELIZABETH RATHER*

27

Michael Ham interviews the president of FORTH, Inc. about the history of Forth and of the first company to deal in Forth systems and applications. She speaks frankly of successes, lean times, and public perception.

---

## *RUN-TIME STACK ERROR CHECKING • BY CHARLES SHATTUCK*

32



System-crashing stack errors make for frustrated students who aren't excited about getting over Forth's learning curve. Run-time stack checking is a good, temporary aid for Forth beginners plagued by mysterious system crashes.

---

## *PERPETUAL DATE ROUTINE • BY ALLEN ANWAY*

34



Ever since humans began numbering days and years, devising a perennially accurate calendar has been a problem. This article presents definitions that calculate the Gregorian day and weekday in fixed-point, double-precision Forth.

---

## *HEADLESS COMPILER • BY DARREL JOHANSEN*

36



The headers of words that are needed only a few times, for definitions of higher-level words, just take up valuable dictionary space. This compiler can be loaded into any Forth-79 system, compiles code with or without headers, and can be forgotten when it is no longer needed.

---

## *EDITORIAL*

4

## *LETTERS*

5

## *ADVERTISERS INDEX*

38

## *FIG CHAPTERS*

42

# EDITORIAL

## **Forth Dimensions**

Published by the  
**Forth Interest Group**  
Volume IX, Number 1  
May/June 1987

*Editor*

Marlin Ouverson  
*Advertising Manager*  
Kent Safford

*Design and Production*  
Berglund Graphics  
ISSN#0884-0822

**W**elcome to a new volume of *Forth Dimensions*. During the past few months, we have thought a lot about the people who use Forth, and the interesting, progressive things they do with it. We felt it was time to bring our publication's physical appearance more in line with its contents. Cynthia Berglund, who does our design and physical production, met with me and advertising manager Kent Safford to discuss a new design. She translated our conceptual abstractions into the tangible pages you hold in your hands. Meanwhile, Kent worked overtime with suppliers, support materials, and advertisers, among other things.

Fortunately, desktop publishing makes this upgrade far less costly than it looks. Thanks to an intensive team effort, our budget has barely budged; but we think the final result looks more pleasing and contemporary. We hope you agree.

A new columnist debuts in this issue. Mitch Bradley is the author of our "Forth to the Future" series. His first installment discusses the importance of 32-bit compatibility to Forth's future, and includes a proposed wordset glossary.

Forth, Inc.'s president, Elizabeth Rather, called with some comments about last issue's "State of the Standard," in which a CBEMA motion for an ANS Forth was reported, along with George Shaw's concerns about that motion and his own IEEE proposal. Ms. Rather found the criticism of the CBEMA proposal inaccurate. She maintains that CBEMA grants its Technical Committees a good deal of latitude in scheduling their

activities, and that voting membership and size of such a committee is not restricted at all (some continuity is required of members). And the proposing body — far from being biased in favor of a single vendor — includes Ray Duncan of Laboratory Microsystems, Don Colburn of Creative Solutions, and several of those major vendors' clients (IBM, MCI, Oak Ridge National Laboratory).

Elizabeth added that participation is open; anyone who wants to be informed about upcoming, Forth-related CBEMA events can contact Elizabeth Rather at Forth, Inc. (Manhattan Beach, CA), or Chris Colburn at Creative Solutions (Rockville, MD).

Since that conversation, Ms. Rather wrote that CBEMA's "...final vote for the establishment of X3J14, the Technical Committee for ANS Forth was favorable: 36-1-1 .... An organizational meeting has been scheduled for August 3-4, 1987 at CBEMA headquarters." A newsletter will be sent to all ANSI members, a press release was to have been mailed on May 1, and the 100+ people on the Forth Vendors Group mailing list will be notified.

If you still haven't had enough about standards, be sure to read the letter in this issue from Guy Kelly, chairman of the Forth Standards Team. And for more about Elizabeth Rather and Forth, Inc., you will find the interesting interview, "Starting Forth, Inc." Enjoy!!

—Marlin Ouverson  
*Editor*

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

Subscription to *Forth Dimensions* is free with membership in the Forth Interest Group at \$30 per year (\$43 overseas air). For membership, change of address, and to submit items for publication, the address is: Forth Interest Group, P.O. Box 8231, San Jose, California 95155. Administrative offices and advertising sales: 408-277-0668.

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

### **About the Forth Interest Group**

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

# LETTERS

## 32 is a Bit Too Many

Dear Mr. Ouverson,

Are we all trapped by power-of-two chauvinism?

There seems to be an impression that the next logical advance in Forth is from a 16-bit word to a 32-bit word. Can we take a moment to consider 24 bits?

One of the beauties of Forth is that addresses and data are interchangeable, since both use 16 bits. Forth's 64K address-space limit is beginning to seem a disadvantage, and its 16-bit single numerical precision is limiting. Expanding both data and address to 32 bits is easy on a 32-bit microprocessor, but I feel that four gigabit addressing and numerical precision is wasted in most applications. Real memories are unlikely to exceed eight megawords for a considerable time, and the ability to individually address every byte on a hard disk is hardly necessary in a block-oriented language. Meanwhile, a 32-bit Forth engine is either going to have twice as many pins and twice as much silicon as a 16-bit engine, or it is going to multiplex data and addresses and slow down.

So try 24 bits. The real estate and pin count has only risen by 50%. The address space has increased to 16 megawords, though I grant that we lose the ability to address bytes directly. Unsigned numerical precision has increased to the equivalent of seven decimal digits. We can even pack a meaningful floating-point number into 24 bits. Double-precision numbers now have 14 decimal digits of precision, which should be enough for most of us.

Food for thought, or just a red herring?

Kind regards,  
Tom Napier  
Dresher, Pennsylvania

## WITHIN Words

Newcomers to Forth from other languages sometimes complain about the relative lack of standardization, particularly the use — by different versions of the standard, and by different implementations of standard and non-standard Forths — of the same word name to mean different things. Some of these complaints are poorly justified.

In my opinion, as a relative newcomer to Forth (a year and a half as a professional Forth programmer, after working in the Pascal world), the changes perpetrated by the Forth-83 Standard are improvements I wouldn't want to live without. But, as with Pascal (which had the "advantage" of being standardized at its inception, rather than after a decade of practical use like Forth), the standard wordset is never enough. Other words are included in different implementations, and they spread through the Forth community if programmers find them useful and memorable. A good name is not the least important feature of such a word.

One useful non-standard, but popular, word is **WITHIN**, which takes three numbers from the stack and returns a flag — that indicates whether the first number is within the range defined by the second and third. The stack picture is (n low high -

-- flag), where the flag is true if and only if  $low \leq n \leq high$ .

In a recent revision of their F83 model implementation, however, Laxen and Perry have changed the meaning of the word **WITHIN**. It retains the same stack picture, but the significance of the flag is changed so that it is true if and only if  $low \leq n < high$ . In other words, where previously the flag was true in the case  $n = high$ , in the new version it is false. Laxen and Perry have kept the old version of **WITHIN**, too, renaming it **BETWEEN**. But this seems, to me, to confuse things even more: as a native speaker of English, I don't see any reason to choose **BETWEEN** as a better name for the old **WITHIN**, nor do I feel any natural affinity for either **WITHIN** or **BETWEEN** as the name of a range test that includes one endpoint and not the other.

As an alternative to the pair of word names **WITHIN** and **BETWEEN**, I would like to introduce a set of four names. Probably nobody, with the possible exception of Wil Baden, would want to implement all of these words (one of them — any one, for that matter — would be enough), but perhaps these names could be taken as something like unofficial "controlled reference words." All of them have the same stack picture as **WITHIN** and **BETWEEN**, and are listed below with the truth condition of the flag:

```
WITHIN low <= n <= high  
WITHIN( low <= n < high  
 )WITHIN low < n <= high  
 )WITHIN( low < n < high
```



# NGS FORTH

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

## STANDARD FEATURES INCLUDE:

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

A COMPLETE FORTH  
DEVELOPMENT SYSTEM.

PRICES START AT \$70

NEW ◀ HP-150 & HP-110  
VERSIONS AVAILABLE



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

My **WITHIN** is the same as the Guy Kelly and original F83 **WITHIN** and the new F83 **BETWEEN.WITHIN** ( is the same as F83's new **WITHIN**. In all cases, an open parenthesis in a word name indicates whether the respective endpoint of the low-to-high number interval is excluded from the range of true values. This convention of open parentheses corresponds to a familiar mathematical convention (many of us learned it with the "new math" a few decades ago) that represents ranges on the real number line. Perhaps the greatest advantage of these word names is that, when you've once seen the stack picture, it is impossible to forget which is which.

If my suggestion meets with the approval of the Forth community, perhaps we can squash **BETWEEN** and the new **WITHIN** before they infect too much written code.

Richard Astle  
La Jolla, California

## A Couple of CASEs

Dear Marlin,

Wil Baden, in his "Ultimate CASE Statement" (FD VIII/5), did an excellent

```

Screen # 11
0 \ CASE
1 : CASE   COMPILER DUP 0 ; IMMEDIATE
2 : !      1+ DUP 1 > IF COMPILER OR THEN COMPILER OVER ;
3 IMMEDIATE
4 : OF     IF COMPILER OR THEN [COMPILER] IF COMPILER DROP
5 ; IMMEDIATE
6 : ENDOF  COMPILER EXIT [COMPILER] THEN ; IMMEDIATE
7 : ENDCASES COMPILER DROP ; IMMEDIATE
8
9 : BREAK  COMPILER EXIT [COMPILER] THEN ; IMMEDIATE
10
11 \ Ed Petsche
12 \ 620 West St.
13 \ Greenport, N. Y. 11944
14
15

```

```

Screen # 12
0 \ CASE TEST
1 : BETWEEN ( n n1 n2 -- ? )
2 1+ OVER - >R - R> U< ;
3 : CRAPS ( n -- )
4 CASE 7 = ! 11 = OF ." you win" BREAK
5 CASE 2 3 BETWEEN ! 12 = OF ." you lose" BREAK
6 . ." is your point" ;
7 : WHATEVER
8 CASE 0= OF ." zero" ENDOF
9 CASE 0< OF ." minus" ENDOF
10 CASE DUP 1- AND 0= OF ." power of 2" ENDOF
11 CASE ASCII 0 ASCII 9 BETWEEN OF ." digit" ENDOF
12 CASE ASCII , ASCII / BETWEEN
13 ! ASCII : = OF ." punctuation" ENDOF
14 ENDCASES ." whatever" ;
15

```

```

Screen # 13
0 \ CASE TEST
1 HEX
2 : CLASSIFY
3 CASE 0 1F BETWEEN
4 ! 7F = OF ." control char" ENDOF
5 CASE 20 2F BETWEEN
6 ! 3A 40 BETWEEN
7 ! 5B 60 BETWEEN
8 ! 7B 7E BETWEEN OF ." punctuation" ENDOF
9 CASE 30 39 BETWEEN OF ." digit" ENDOF
10 CASE 41 5A BETWEEN OF ." upper case" ENDOF
11 CASE 61 7A BETWEEN OF ." lower case" ENDOF
12 ENDCASES ." not a character" ;
13
14
15

```

job describing problems with the many case proposals that have been published to date. He also stated requirements for further proposals and gave his own version of a sweetened case statement. After reading the article and looking at his examples, **CLASSIFY** in particular, I felt that **CASE** could use just one more lump of sugar. The stack operators and redundant **ORs** should be eliminated, if possible. My method for eliminating them is to use **!** to separate the tests within a case clause. This separator is also used to hide the stack and **OR** operations. **ENDCASES** is used as a synonym for **DROP**. It is optional, and is not used when the default code needs the value on top of the stack. Also, the **EXIT THEN** sequence at the end of each case clause is compiled as a macro (**ENDOF**).

I thought **BREAK** might be a better name for the **ENDOF** macro, and I've included it in one of the examples. One of the advantages of using **BREAK** is that it could be used outside of the case statement, to replace other **EXIT THEN** sequences.

It would be great to see an article similar to Wil Baden's case article, but discussing requirements for Forth database elements. Also, I hope there will be more articles on classes in future issues.

Ed Petsche  
Greenport, New York

Dear Marlin:

I'm about "cased out," so when I saw not one, but two articles on case statements in the last issue, I proceeded

with reluctance. I have only this to say about Wil Baden's Ultimate Case: if there was ever a subject about which nothing is ultimate, surely it is the case statement!

Happily, Allen Anway's "Tinycase" provided something to think about. Here is something a little different, and with a twist I sort of liked. In this case (is that a pun?), there is a more extended comment I'd like to make.

Allen's statement that, "It takes some stack gymnastics for the high-level word to work out..." is an understatement. I haven't seen such convoluted stack maneuvering since I last looked at some of the first Forth code I ever wrote. See the enclosed screens for an example of how to do it with no stack gymnastics.

Allen's original example becomes:

```
4 TRANSCASE TEST1
2,234,7,789,18,181,97,
979, ( nocase = default )
```

As can be seen, the only difference is that the default is not entered as part of the case-translation entries.

By the way, the **BEGIN WHILE UNTIL** (Anway screen 80) combination will give some Forth compilers trouble (e.g., my CP/M F83 version 2.0.1). Anyway, it seems to me that both a mid-pass test and a post-pass test for completion of an indefinite loop isn't needed. It was this problem that actually got me started on the rewrite, resulting in the enclosed screens.

Sincerely,  
Gene Thomas  
Little Rock, Arkansas

```

1
0 \ Transcase -- see A. Anway, FD VIII/5, p. 23      Jan1987:gt \S                               Feb1987:gt
1 FALSE CONSTANT NOCASE                               NOCASE           A constant to leave a default if no case match
2 : TRANSCASE ( cnt -- :definer ) CREATE DUP @, DOES) TRANSCASE Defining word -- precede with count of cases
3   DUP @ SWAP 2+ SWAP                                TRANSCASE       Get count and incr to first case addr
4   DEPTH 3 - >R      (S kb-entry addr cnt -- )      Save depth relative to 0
5   BOUNDS DO                                           FB3 :bounds over + swap ; start loop
6     DUP                                               DUP the key board entered value and
7     I @ =                                             compare it with the case value at addr I and
8     IF                                               if it matches leave the translation on the stack
9     I 2+ @ SWAP LEAVE                                under the KB entered value (to be dropped)
10    THEN                                             The 4 +LOOP explains squared count (DUP @) above
11    4 +LOOP DROP                                     incr past translation to next case-- drop KB entry
12    DEPTH R> =                                       DEPTH changed?
13    IF                                               If so, translation in on the stack
14    NOCASE                                           If not, then leave the default
15    THEN ;

```

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

HS / FORTH

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

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

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

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



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

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



Find Out How To Implement  
Real-Time Systems In:

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

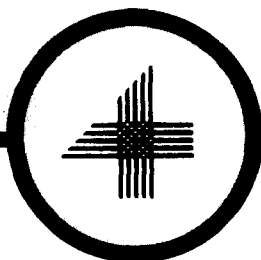
... on time and under budget.

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

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

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

**ON TIME.  
UNDER BUDGET.**



**FORTH, Inc.**

By the way, the **BEGIN WHILE UNTIL** (Anway screen 80) combination will give some Forth compilers trouble (e.g., my CP/M F83 version 2.0.1). Anyway, it seems to me that both a mid-pass test and a post-pass test for completion of an indefinite loop isn't needed. It was this problem that actually got me started on the rewrite, resulting in the enclosed screens.

Sincerely,  
Gene Thomas  
Little Rock, Arkansas

## FST Accepts Proposals

Dear Marlin,

Thanks for the fine job you're doing at *Forth Dimensions*. Perhaps your readers might be interested in some thoughts about a few of the events at the 1986 FORML Conference.

There were a variety of working groups again this year, and one that I attended discussed both the possible creation of an ANS Forth and the activation of the Forth Standards Team's (FST) technical proposal phase. There was also a presentation (by a small panel) concerning a request made to CBEMA to sponsor a Forth standards effort.

Having been a member of both the working group and the panel, I discussed my impressions of them with the local FIG Chapter, and also with a variety of other members of the Forth user community (including several FST members). My feeling now is that a majority of Forth users are in favor of an ANSI Forth Standard and that they would like the FST to begin its technical proposal phase.

There seemed to be general agreement that (1) any ANSI effort should be to formalize existing practice and should be limited to resolving any restrictions and any machine or operating system dependencies of the Forth-83 Standard, and (2) any FST effort should be to address extensions to the standard.

Proposed issues to be addressed by any ANSI effort would be:

Arithmetic and logical operators  
Flow-of-control structures  
Input and output operators  
Memory and mass-storage operators  
Exception handling  
Vectored execution  
Compiler-extension operators  
Data-description operators  
ROM-based applications

Proposed extensions to be addressed by FST technical proposals might include:

Floating-point operators  
String operators  
Communications operators  
Graphics operators  
Data-structure operators  
File operators  
Operating-system-interface methods  
Multi-tasking methods

There continues to be discussion as to whether it would be better to proceed toward an ANS Forth via CBEMA or IEEE (or even via a joint effort between the two). There is, unfortunately, also a feeling on the part of some members of the Forth community that something could go wrong. Some comments implied that:

- The Forth Standards Team, working in secret, produced a minority document.
- The group that submitted the request for an ANS Forth worked in secret and will produce a minority document.
- The CBEMA group will work in secret (or will cost too much to participate in) and will produce a minority document.
- All of the above groups have done, or are going to do, something that was, or will be, undesirable.

My responses to these comments are:

Having attended and worked at all of the Forth Standards Team meetings since the 1979 Catalina meeting (which I attended as a non-voting observer), I can assure anyone that there was no intent on the part of the FST to be secretive or to produce something undesirable. All meetings were publicized well in advance, observers were welcomed and encouraged to participate in discussions and straw votes, the meeting results were reported



and discussed in *Forth Dimensions*, and a draft proposed standard was published and distributed.

The FST members were (and are) dedicated to advancing the Forth language and its use. I can understand debate as to the quality and desirability of the various standards, but am puzzled by questions regarding the intent of the group.

The group that submitted the request to CBEMA was (and is) composed of users, vendors, and FST members. They have made copies of the submittal available to any interested party (I asked for a copy, and have received and read it). They also seem dedicated to furthering the cause of Forth.

A CBEMA working group may cost more to participate in than an IEEE group or the FST, but it takes a great deal of dedication of both time and money to travel to and attend standards meetings, no matter who sponsors them. The best most of us can hope for is that all meetings will be open to public observation, that the proceedings of the meetings will be widely published, and that input from the Forth community will be seriously considered. If so, then each of us can help determine the quality of any resulting Forth standard.

Please, let's all work together to support those who have the time and energy to attempt to enhance the status of Forth. If you can't directly participate in the formal efforts, you can follow and respond to the published standards committee reports, proposals, and draft standards. Also, remember the other avenue open for input by the Forth community, the FST technical proposal.

As a result of input from FORML and elsewhere, the FST is soliciting technical proposals. The technical proposals should be of two general categories:

1. Proposed extensions to the Forth-83 Standard.

These should be as generally useful as possible, and should attempt to conform to practices currently accepted by the majority of the Forth community. The intent should be to codify proven techniques, not to generate new and

untried approaches (submit those to FORML). The extensions should not be considered as mandatory additions to the standard, but as optional add-ons. Their main purpose should be to allow communication of techniques in a standard, transportable format. They should be more useful for the transport of technology between programmers than for incorporation into all commercial applications. They should be more concerned with defining commonly accepted names for standard operations than with locking a vendor or user into a single method of accomplishing a task.

2. Modification of the existing Forth-83 Standard.

These should only be submitted if it is necessary to remove ambiguities or to generalize the standard in order to incorporate the standard extensions. They should not be attempts to "fix" an offending word or to change the intent of the existing standard. They should provide input to, or should attempt to conform to, any ANSI effort.

The forms and procedures for preparing technical proposals are published in the Forth-83 Standard. The following plan has been adopted to help with the creation and publication of submittals:

1. Become involved with a particular effort. Select one or more topics of interest. Contact others in your area who are interested in the topics. Contact the FST and get on the appropriate interest lists. The FST will attempt to place all interested parties in contact. One of the interested parties should become the group leader for the topic.

2. Prepare and distribute preliminary copies of the proposal. Copies should be sent to any person(s), publication, or bulletin board the working group deems appropriate. Telecommunication should also be considered.

3. Submit a final version to the FST. The FST will collect all the proposals for yearly publication.

## FORTHkit

5 Mips computer kit

\$400

### Includes:

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

### Instructions:

Easy assembly  
cmFORTH listing  
shadows  
Application Notes  
Brodie on NC4000

### You provide:

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

### Supports:

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

### Inquire:

**Chuck Moore's  
Computer Cowboys**

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

Interested parties should, at any time, be able to obtain copies of proposals from the appropriate working groups or from the FST. The FST plans to publish updated lists of working groups, with the name, address, and telephone number of the group leader, in *Forth Dimensions*.

An additional, planned activity of the FST is to solicit technical information and documents from each Forth vendor, for review and comparison with the techniques and methods proposed by the working groups. This will provide additional input concerning currently accepted practices. Any vendor that wishes to participate in the sharing of such material should send the documents to:

Forth Standards Team  
P.O. Box 4545  
Mountain View, California 94040

or to:

George W. Shaw II  
P.O. Box 3471  
Hayward, California 94540

Vendors with questions should contact George Shaw at 415-276-5953. Other questions should be sent to the FST address, or call me evenings or weekends at 619-454-1307.

Sincerely,  
Guy M. Kelly, FST Chairman  
La Jolla, California

## DASH, FIND & ASSOCIATES

Our company, DASH, FIND & ASSOCIATES, is in the business of placing FORTH Programmers in positions suited to their capabilities. We deal only with FORTH Programmers and companies using FORTH. If you would like to have your resumé included in our data base, or if you are looking for a FORTH Programmer, contact us or send your resumé to:

DASH, FIND & ASSOCIATES  
808 Dalworth, Suite B  
Grand Prairie TX 75050  
(214) 642-5495



Committed to Excellence

### Final Exam: F101 ;

Name:

Date:

#### Part 1 — Subroutines

1. Define an algorithm to produce an output that is a reverse mirror image of itself. Find and justify 10 political applications for the algorithm.

2. Write a random-number generator to output an infinitely non-repeating series; do not use any conditionals. Construct a simulation to prove its efficacy for predicting the outcome of everything.

#### Part 2 — History and Philosophy

3. Explain in detail the philosophy of Forth as it relates to Fortran, COBOL, Pascal, Ada, LISP, C, and the morality of nuclear war. Give an example in each language, with a translation to BASIC, and note specific differences and similarities between Forth and nuclear war.

4. Outline the history of Forth, including each commercial and public-domain version of each standard. Draw a comparison of your outline with the history of the world. Be concise.

#### Part 3 — Practical Application

5. Write a program to prove the big-bang theory, including a routine to produce a big bang, creating a universe in the place of your choice. Do not expend more than 5 amps.

6. Prepare a thesis in psuedo-code on the topic, "Real-time control of rioting crowds with Forth." Be prepared to defend your thesis. A rioting crowd will be provided.

Gene Thomas  
Little Rock, Arkansas

## 80386 for PC

\$29.95 buys:

- Schematic
- Timing diagram
- Parts list
- Where to buy
- Board layout
- Descriptions of:
  - Hardware
  - Interface
- Wirewrap instructions
- Bibliography
- Forth screen

Use low-cost 6264 static rams, or 43256's for 256K

Build your own Intel 80386 co-processor board in a few evenings. This kit includes the complete design for a Wire-wrap project, including all pin-outs. The 80386 uses 8Kx8 or 32Kx8 memory chips in parallel for maximum speed, yet the 8bit PC bus can access the whole memory array thru a 32K window.

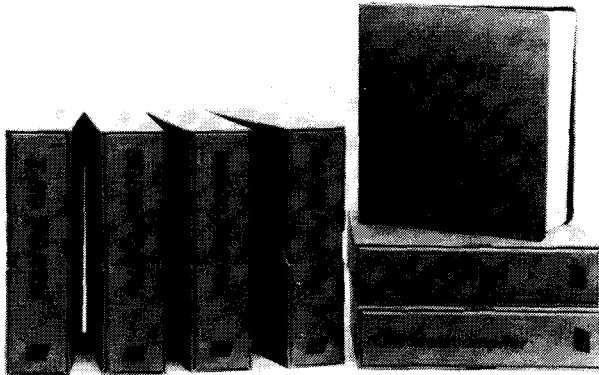
Selectable window -- works with any PC/XT/AT display  
The PC can Reset, Hold, or interrupt the 80386  
The PC can poll one 8bit port while the 80386 runs full speed  
When held, the PC has access to ALL of the 80386 memory  
Use Forth or DEBUG to assemble, control the 80386, view memory

Don't have a PC? Adapt the design to any 8-bit bus with these four strobes: memory read, write, I/O port read, write.

Send \$29.95 (check or money order) to Hampton Corporation  
20800 Center Ridge Road  
Cleveland, Ohio 44116

Immediate delivery

# TOTAL CONTROL with LMI FORTH™



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

### For Development: Interactive Forth-83 Interpreter/Compilers

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

### For Applications: Forth-83 Metacompiler

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

### For Speed: CForth Application Compiler

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

### Support Services for registered users:

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

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

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

#### Overseas Distributors.

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

(Continued from page 15.)

the appropriate graphics-mode initialization word on line 9 of screen 28 you should be able to run this program with any graphics display. Playing with the scale factor on screen 18 and/or the horizontal scale factors commented out in screen 21 may make the pictures more pleasing on some displays.

### References

1. *Fractals: Form, Chance, and Dimension*, B.B. Mandelbrot. W.H. Freeman and Company, San Francisco, 1977.

### Recommended Reading

"3-D Fractals," M. van de Panne. *Creative Computing*, July 1985, pp. 78-82.

*Art of Computer Programming*, D.E. Knuth. Addison-Wesley Publishing, Vol. 2, 1981 (includes a discussion of random number generators).

"Fractals," P.R. Sorensen. *BYTE*, September 1984, pp. 157-172.

*The Fractal Geometry of Nature*, B.B. Mandelbrot. W.H. Freeman and Company, San Francisco, 1983.

"IBM Fractal Graphics," P.W. Carlson. *Compute!*, March 1986, pp. 78-80.

*Phil Koopman, Jr. is the vice-president and chief engineer for WISC Technologies. The program discussed in this article was originally developed as a demonstration for the WISC CPU/16.*

FORTH-79

# FRACTAL LANDSCAPES

PHIL KOOPMAN, JR.- NORTH KINGSTOWN, RHODE ISLAND

In the last issue of *Forth Dimensions*, I presented a program that used the Bresenham algorithm to draw lines on a computer graphics display. Now I will build upon those definitions to create a program that draws pseudo-randomly generated fractal landscapes with height-based coloring (including a sea-level) and hidden-surface elimination.

### What's a Fractal?

The word fractal is short for "fractional dimension," and is used to describe various geometrical shapes that are not strictly one, two, or three-dimensional objects. Interestingly enough, most naturally occurring objects (such as coastlines) and natural phenomena (such as Brownian motion) are best described by fractal geometry<sup>1</sup>.

To get a better feel for what a fractal is, consider the western coastline of the United States. From far away in space, the coastline looks more-or-less straight; that is, it looks like a one-dimensional line having a fractional dimension near 1. As an observer gets closer, the perceived length of the coastline and its fractional dimension increases as more details such as inlets and peninsulas become visible. In fact, with a very close look at an area like San Francisco Bay, the coastline becomes a two-dimensional object with a fractional dimension near 2.

### How To Draw Random Fractals

To use the fractal concept to draw a landscape, let's first look at how a fractal can be used to draw a shape, starting from a line. We will look at a random fractal line, but of course there are other ways to draw fractal lines (see the recommended reading list).

```
SCREEN #15
0 \ Special 32-bit unsigned multiply
1 DECIMAL
2 \ This is a special-purpose unsigned multiply that returns only
3 \ the low-order 32 bits. For a more generalized multiply,
4 \ see: P. Koopman, MVP-FORTH INTEGER & FLOATING POINT MATH
5 \ MVP-FORTH series Vol. 3 (revised), Mountain View Press
6
7 : XD* ( UD1 UD2 -> UD3 )
8 OVER 5 PICK U* >R >R
9 4 ROLL U* DROP >R
10 U* DROP
11 0 SWAP R> 0 SWAP D+
12 R> R> D+ ;
13
14
15
```

```
SCREEN #16
0 \ 32-BIT BASED LINEAR CONGRUENT RANDOM NUMBER GENERATOR
1 DECIMAL \ From Knuth's Art of Computer Prog. vol. 2, page 170
2 DARIABLE SEED \ High 16-bits of SEED are actual random #
3 \ Store all 32-bits initially to re-seed generator
4 3141592653. SEED D!
5
6 : RNDF ( -> N )
7 SEED D@ 3141592621. XD* 1. D+
8 DUP ROT ROT SEED D! ;
9
10 \ The below random number generator is very poor, but
11 \ produces interesting smoothed/rounded rolling hills
12 : RNDF ( -> N )
13 \ [ SEED 1+ ] LITERAL @ 31417 U* 1. D+ DUP ROT ROT SEED D! ;
14
15
```

```
SCREEN #17
0 \ FRACTAL LAYOUT
1 DECIMAL
2 \ SQUARE P1+----x-----+P2
3 \ LAYOUT: ! I ! II ! Roman #'s are square #'s
4 \ x----x-----x P1..P4 are corner point #'s
5 \ ! IV ! III !
6 \ P4!----x-----!P3
7 \ In the recursive routines, P1..P4 are the address of the
8 \ points within the layout array
9 \ Each recursion calculates the new "x" point height values
10 : WALL ; \ Useful point for FORGETting
11 : RECURSE
12 LATEST PFA CFA , ; IMMEDIATE
13
14 : CELL* 2* ; \ Change to fit bytes/cell of your machine
15 : CELL+ 2+ ; \ Change to fit bytes/cell of your machine
```

# HARNESS THE POWER OF FORTH!



## FORTH: A TEXT AND REFERENCE

*by Mahlon G. Kelly and Nicholas Spies*

The syntax, powerful architecture, and program design of FORTH make it **the most progressive way to dramatically increase software productivity.**

**Both a highly approachable text and an all-inclusive, easily-accessible reference,** this 512 pp., handbook on FORTH accommodates those with no prior computer knowledge as well as advanced programmers looking to delve more deeply into the language. It is designed for both classroom use and self-study.

**This definitive guide to the FORTH language offers:**

- a more extensive approach that starts with, but goes far beyond, the introductory level.
- coverage of both the 1979 and 1983 standards
- sample coverage of a typical advanced implementation, MMSFORTH
- very extensive examples and exercises with complete answers
- very complete index with more than 1500 entries and a detailed nine page table of contents
- accessible cross-referencing and 68 pages of separate FORTH words and terminology glossaries make the book an outstanding desk reference
- and extended program, a generic screen editor, is used to explain the unique program-design procedures of FORTH *January 1986. Paper \$21.95. Cloth \$28.67*



## STARTING FORTH, 2nd Edition

*by Leo Brodie*

Considered to be the **"bible"** of the introductory books on FORTH, this all-inclusive presentation of the FORTH programming language and operating system allows the beginner a clear, effective understanding of the whole FORTH system.

**Praised as one of the best programming language books in the industry,** this re-examined edition presents:

- the FORTH-83 standard systems
- differences from the popular fig-FORTH model
- the effective coding style described in Thinking FORTH.
- an alphabetical index to glossary terms
- increased self-study problems
- special footnotes that address FORTH-79, MVP-FORTH and fig-FORTH versions

Each outstanding chapter concludes with a "Review of Terms", practical study problems, "Handy Hints", and features three detailed examples of FORTH in application.

*Prices subject to change without notice*

*September 1986. Paper \$19.95. Cloth \$26.67*



  
**PRENTICE HALL**  
Englewood Cliffs, NJ 07632

SIMON & SCHUSTER HIGHER EDUCATION PUBLISHING GROUP

*Available at finer bookstores or direct from Prentice Hall, (201) 767-5937*

First, consider the straight line segment shown in Figure One-a. We will take the midpoint of that line segment and pull it to one side, as shown in Figure One-b. Next, we will recursively take each midpoint of the resulting line segments and pull them randomly to one side or the other as shown in Figures One-c and One-d. As you can see, this process quickly results in a wandering line. For the most pleasing shape, the amount of "pull" applied is cut in half at each level of recursion, forming a smooth result.

In order to extend this concept to an area instead of a line, the "Landscape" program on screens 15-28 forms a two-dimensional array. Each cell in the array holds the height of a point above or below sea level. The word **CALCULATE-SERVICE** recursively breaks this array into smaller and smaller squares, using the addresses of the four corners of the array instead of four pairs of (X,Y) coordinates. **SET-HEIGHTS** sets the heights for the array cells at the midpoints of the sides of the current square and for the center of the current square, then breaks the square up into four sub-squares (see the diagram on screen 17 for a description of the nomenclature used by the subdividing algorithm). After the data array has a height associated with each point, the program uses the **SEA-LEVEL** word to reassign all negative heights to sea level.

After the heights are computed, the landscape is drawn on the screen. As each point of the array is drawn, it is assigned a color based on height and the number of colors available.

### Screen-Drawing Tricks

I have used several tricks in "Landscape" to speed up the screen-drawing time. This drawing time would be prohibitively long if conventional, three-dimensional graphics techniques were used.

The most time-consuming part of many graphics drawing programs involves 3-D transformations, especially rotations. On the other hand, a top or side view of a fractal landscape would not be terribly exciting. I solved this speed-versus-prettiness dilemma using two techniques: a "sleazy" rotation to elevate the rear of the picture, and an

```
SCREEN #18
0 \ FRACTAL DATABASE
1 DECIMAL
2 5 CONSTANT #LEVELS \ Number of recursion levels
3 65 CONSTANT SIZE \ array size = 1 + 2**(#LEVELS+1)
4 \ NOTE: Change SIZE to 129 and #LEVELS to 6 for EGA
5 \ SQUARE-P1 is a 2D array that holds heights of all grid points
6 CREATE SQUARE-P1 SIZE SIZE * CELL* ALLOT
7 SIZE 1- CELL* SQUARE-P1 + CONSTANT SQUARE-P2
8 SIZE SIZE * 1- CELL* SQUARE-P1 + CONSTANT SQUARE-P3
9 SIZE SIZE 1- * CELL* SQUARE-P1 + CONSTANT SQUARE-P4
10
11 : SCALE 2* 2* ; \ Scale value of pixels per data array point
12 1 SCALE CONSTANT DELTA
13 : AVE ( U1 U2 -> UAVE ) \ unsigned average of 2 addresses
14 \ NOTE: This is a prime candidate for machine code speed-up!
15 0 SWAP 0 D+ 2 U/MOD SWAP DROP ;
```

```
SCREEN #19
0 \ SPECIAL LINE DRAWS FOR FRACTAL LANDSCAPES
1 DECIMAL
2 : Y-CONVERT ( HEIGHT Y1 -> Y2 )
3 \ For now, assume tilted up 30 degrees in back, no X change
4 \ Inputs are x/y data points & height, outputs screen coords
5 + 2/ NEGATE YMAX + ;
6 : F-MOVE ( X HEIGHT Y-INDEX -> )
7 \ Use the code on the next line for tracing if desired
8 \ ." MOVE:" SWAP U. U. CR ?TERMINAL ABORT" F-MOVE" ;
9 SCALE Y-CONVERT MOVE-CURSOR ;
10 : F-LINE ( X HEIGHT Y-INDEX -> )
11 \ Use the code on the next line for tracing if desired
12 \ ." LINE:" SWAP U. U. CR ?TERMINAL ABORT" F-LINE" ;
13 SCALE Y-CONVERT DUP 0<
14 IF ( Clip line that is off screen ) DDROP
15 ELSE LINE THEN ;
```

```
SCREEN #20
0 \ INITIALIZE THE HEIGHT ARRAY & CALCULATE COLOR FOR A HEIGHT
1 HEX
2 : INITIALIZE-SQUARE ( -> )
3 \ Fill all initial heights with 8181 for a recognizable tag
4 SQUARE-P1 SIZE 0
5 DO DUP SIZE CELL* 81 FILL SIZE CELL* + LOOP DROP
6 20 SQUARE-P3 ! \ Initial values to slant landscape
7 18 SQUARE-P4 ! \ "forward" for a better view
8 -15 SQUARE-P1 ! -10 SQUARE-P2 ! ;
9 : SET-COLOR ( HEIGHT -> ) \ Figure color for given height
10 \ Adjust the "40" on the line below to individual taste.
11 \ In particular, change to a "18" for EGA
12 DUP 8 < IF ( near sea level ) DROP 1
13 ELSE 40 / #COLORS 2- MOD 2+ THEN COLOR ! ;
14 \ Redefine as : SET-COLOR DROP 1 COLOR ! ; for CGA/HIRES
15 DECIMAL
```

```
SCREEN #21
0 \ DRAW THE HEIGHT ARRAY ON THE CRT DISPLAY
1 DECIMAL
2 : DRAW-SURFACE ( -> ) \ Draw from bottom to top on screen
3 SIZE 2- 0 DO ( column ) I SIZE + CELL* SQUARE-P1 +
4 10000 ( initial min Y value ) SIZE 1- 1 DO ( row )
5 \ Test for hidden surface removal
6 OVER @ I SCALE Y-CONVERT DDUP >
7 IF ( new min y value means visible point ) SWAP DROP
8 \ Add a 2* where indicated when using CGA/HIRES mode
9 OVER @ SET-COLOR J SCALE ( 2* )
10 DUP DELTA ( 2* ) + 4 PICK SIZE CELL* - CELL* @ I 1- F-MOVE
11 DUP 4 PICK @ I F-LINE
12 DELTA ( 2* ) + 3 PICK SIZE CELL* + CELL* @ I 1+ F-LINE
13 ELSE ( hidden ) DROP THEN
14 SWAP SIZE CELL* + SWAP 1 /LOOP
15 ?TERMINAL ABORT" BREAK" DDROP 1 /LOOP ;
```

unconventional point-connection scheme to eliminate the need for spinning the picture.

A standard rotation of a landscape to elevate the rear of the picture involves using the equation:

$$\text{NEWY} = \text{yvalue} * \sin(\text{angle}) + \text{height} * \cos(\text{angle})$$

for each height data point in the landscape array. In order to eliminate the scaled integer or floating-point arithmetic involved, I chose my rotation angle to be 30 degrees and changed the "\* sin(angle)" term to a division by two. Then, to get rid of the cosine term, I decided to approximate  $\cos(30)=.866$  by 0.5 (division by two) and increased the original height value on line 7 of screen 28 to compensate. The elevation using this strategy is accomplished by **Y-CONVERT** on screen 19.

Even with the rear of the picture elevated, the result is pretty unexciting if points are connected by columns and rows. You would only see regularly spaced vertical lines with landscape profile lines wiggling horizontally across the screen. In order to fix this, lines 10 through 12 of screen 21 connect points in sideways "V" patterns to form a picture composed of diagonal lines instead of mostly horizontal and vertical lines. The lines are drawn and colored by columns of points, front to back.

It turns out that hidden-surface elimination, a major computational drain on many graphics programs, comes at almost no charge when using the drawing technique described above. Since points are drawn from front to back, lines 5 - 7 of screen 21 simply ensure that each new Y value for a point to be displayed is further up on the screen than any previous Y values for that column.

### Running The Program

Simply type **LANDSCAPE** from the Forth "OK" prompt. The program will draw a landscape and wait for a keystroke on a PC-compatible machine with a Color Graphics Adapter (CGA) display. If you change the constants on lines 2 - 3 of screen 18, redefine the coloring word on lines 9 - 14 of screen 20, and substitute

*(Continued on page 11.)*

```
SCREEN #22
0 \ SET-HEIGHT -- Set height of a point for recursive processing
1 HEX
2 : SET-HEIGHT ( DH LEVEL PX VALUE PY VALUE -> DH LEVEL )
3   ROT + 2/ ROT ROT AVE
4   DUP @ 8181 =
5   IF ( store ) SWAP 4 PICK RNDP +- + SWAP !
6   ELSE DDROP THEN ;
7
8 DECIMAL
9
10
11
12
13
14
15
```

```
SCREEN #23
0 \ SET HEIGHTS FOR ALL THE "x" POINTS TO MAKE SUB-SQUARES
1 DECIMAL
2 : SET-HEIGHTS ( P1 P2 P3 P4 DELTA-H LEVEL# -> <unchanged> )
3 \ Following 2 lines are debug/trace code to watch recursion
4 \ CR 6 PICK U. 5 PICK U. 4 PICK U. 3 PICK U. OVER U. DUP U.
5 \ ?TERMINAL ABORT" SET-HEIGHTS"
6 ( ave P1/P2 ) 6 PICK DUP @ 7 PICK DUP @ SET-HEIGHT
7 ( ave P2/P3 ) 5 PICK DUP @ 6 PICK DUP @ SET-HEIGHT
8 ( ave P3/P4 ) 4 PICK DUP @ 5 PICK DUP @ SET-HEIGHT
9 ( ave P1/P4 ) 6 PICK DUP @ 5 PICK DUP @ SET-HEIGHT
10 ( ave P1/P3 ) 6 PICK DUP @ 6 PICK DUP @ SET-HEIGHT ;
11
12
13
14
15
```

```
SCREEN #24
0 \ WORD TO SET UP PARAMETERS FOR SUB-SQUARES # 1-2
1 DECIMAL
2 : SQUARE1 ( P1 P2 P3 P4 DELTA-H LEVEL# -> <2.copies> )
3   6 PICK DUP 7 PICK AVE
4   OVER 7 PICK AVE 9 PICK 7 PICK AVE
5   6 PICK 2/ 6 PICK 1- ;
6
7 : SQUARE2 ( P1 P2 P3 P4 DELTA-H LEVEL# -> <2.copies> )
8   6 PICK 6 PICK AVE 6 PICK
9   DUP 7 PICK AVE OVER 7 PICK AVE
10  6 PICK 2/ 6 PICK 1- ;
11
12
13
14
15
```

```
SCREEN #25
0 \ WORD TO SET UP PARAMETERS FOR SUB-SQUARES # 3-4
1 DECIMAL
2 : SQUARE3 ( P1 P2 P3 P4 DELTA-H LEVEL# -> <2.copies> )
3   6 PICK 5 PICK AVE 6 PICK 6 PICK AVE
4   6 PICK DUP 7 PICK AVE
5   6 PICK 2/ 6 PICK 1- ;
6
7 : SQUARE4 ( P1 P2 P3 P4 DELTA-H LEVEL# -> <2.copies> )
8   6 PICK 4 PICK AVE 6 PICK 5 PICK AVE
9   6 PICK 6 PICK AVE 6 PICK
10  6 PICK 2/ 6 PICK 1- ;
11
12
13
14
15
```

*(Screens continued of next page.)*

```

SCREEN #26
0 \ RECURSIVE PROCEDURE TO SET HEIGHTS FOR RANDOM 3-D TERRAIN
1 DECIMAL \ BASED ON SUB-DIVIDED SQUARE FRACTALS
2 : CALCULATE-SURFACE ( P1 P2 P3 P4 DELTA-H LEVEL# -> )
3   SET-HEIGHTS
4   DUP   ?TERMINAL ABORT" BREAK IN CALCULATE-SURFACE"
5   IF ( non-zero level )
6     SQUARE1 RECURSE
7     SQUARE2 RECURSE
8     SQUARE3 RECURSE
9     SQUARE4 RECURSE
10  THEN
11  DDROP DDROP DDROP ;
12
13

```

```

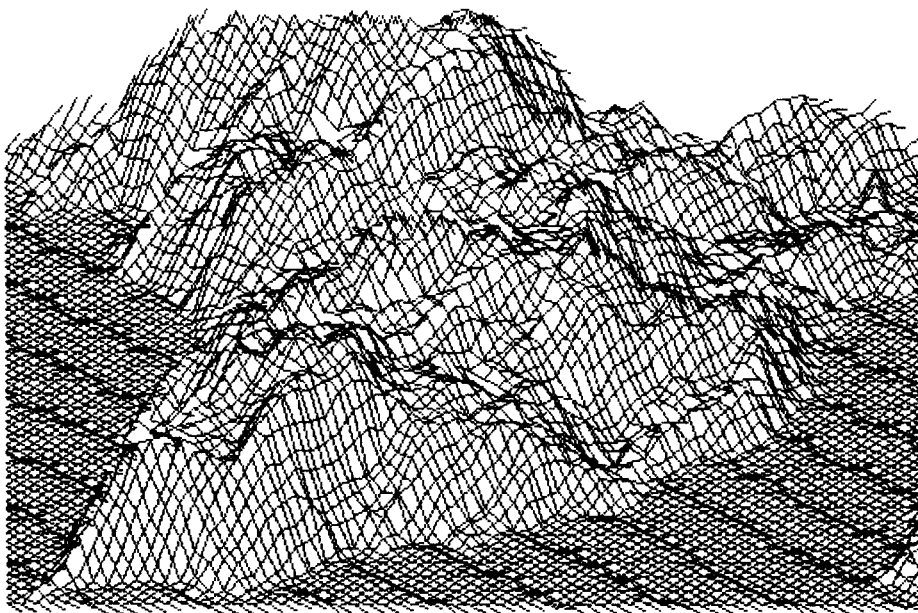
SCREEN #27
0 \ SEA-LEVEL -- SET SEA LEVEL FOR NEGATIVE HEIGHT POINTS
1 DECIMAL
2 : SEA-LEVEL ( -> )
3   SQUARE-P1 SIZE 0 DO
4     SIZE 0 DO
5       DUP @ DUP 0<
6       IF ( below sea level -- add fudge factor for waves )
7         1 AND I J + + 7 AND OVER !
8       ELSE DROP THEN
9     CELL+ LOOP
10  LOOP DROP ;
11

```

```

SCREEN #28
0 \ MASTER PROCEDURE TO DRAW A RANDOM 3-D FRACTAL
1 DECIMAL \ BASED ON SUB-DIVIDED SQUARES
2 DARIABLE SEED-SAVE \ Saves random seed. Placing the saved
3 \ value back into SEED will re-create the same landscape
4 : LANDSCAPE ( -> )
5   SEED D@ SEED-SAVE D! INITIALIZE-SQUARE
6   CR ." Computing new heights"
7   SQUARE-P1 SQUARE-P2 SQUARE-P3 SQUARE-P4
8   YMAX 2/ #LEVELS CALCULATE-SURFACE
9   CR ." Computing sea level" SEA-LEVEL
10  SET-CGA-MODE \ Change to SET-EGA-MODE or SET-CGA-HIRES-MODE
11 \ SEED-SAVE D@ CR ." SEED=" D. ( Optional SEED display )
12  DRAW-SURFACE
13  CR ." Press any key to continue" KEY DROP
14  SET-TEXT-MODE ;
15

```



# BRYTE FORTH

*for the*

# INTEL 8031 MICRO- CONTROLLER



## FEATURES

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

## COST

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

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

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



# FORTH TO THE FUTURE

MITCH BRADLEY-MOUNTAIN VIEW, CALIFORNIA

**32**-bit machines are here to stay. Over the next few years, 32-bit machines will grow in importance. Forth must be able to use the full power of 32-bit machines.

This article presents a consistent, proven scheme for using Forth on 32-bit machines, based on several years of experience with 32-bit systems. It does not address the problems of simulating extended addressing on the 8086. The focus is on making the transition from 16-bit Forth systems to "real" 32-bit architectures like the 80386, the 68000, and the IBM RT.

## Goals

1. Programs should run unchanged on either 16-bit or 32-bit machines.
2. The 32-bit machine must not be penalized. The full power of the 32-bit machine must be available.

## Tradeoffs

1. Existing programs may have to be modified in order to make them run on either size machine.
2. A lot of new words are specified. These words are necessary because the existing words do not work right on 32-bit machines.

## Justification

Forth will not succeed if it remains stuck at 16 bits while the world switches to 32 bits. Insisting that existing programs run unchanged on 32-bit machines penalizes the 32-bit implementation.

The wordset presented here penalizes

neither 16-bit nor 32-bit machines. It adds no new functionality, it simply specifies a set of names for words whose behavior is independent of the machine size.

## What is a 32-bit Machine?

The distinguishing factor is the size of the address arithmetic. The address arithmetic determines the size of an address that can be easily calculated. The 68000 is a 32-bit machine — even though its data path is only 16 bits wide, and even though the package has only 24 address pins — because it is easy to calculate 32-bit addresses. The 80286 is a 16-bit machine, even though it has more than 16 address pins; addresses outside a 16-bit bank are painful to calculate.

Forth prefers to represent an address as a single entry on the stack, since the same operators are used for both number arithmetic and address arithmetic. It is possible, but troublesome, to represent addresses as multiple stack items. The preferred width of the Forth stack on a particular machine is the size of that machine's address arithmetic.

## Compatibility Problems

When moving code from a 16-bit Forth implementation, there are two major problems.

Most Forth programs contain lots of things like `2+` and `6+`. This is fine if you are trying to add two or six to a number, but it causes problems if you are trying to increment an address to point to the next number. On most 32-bit machines, successive numbers are four addresses apart, not two.

16-bit numbers are inadequate for many purposes, so Forth has "double

numbers," which are 32 bits, represented as two stack items. 32-bit systems do not need two stack items to represent a 32-bit number. Existing 16-bit programs use fancy stack manipulations to move the separate halves of a 32-bit number. Double-number operators like `2DUP` are used both for pairs of single numbers and for 32-bit numbers. A 32-bit number is an entirely different thing than a pair of numbers. They just happen to have a similar representation on a 16-bit system. On a 32-bit system, this isn't true.

## Solutions

Some brief words about the nomenclature used here:

A "normal" is a number that is represented as one stack entry. On a 16-bit machine, a normal is 16 bits. On a 32-bit machine, a normal is 32 bits. The majority of all Forth operations are performed on normal numbers.

A "longword" is always a 32-bit number. On a 16-bit machine, a longword is represented as two stack entries. On a 32-bit machine, a longword is represented as one stack entry.

A "word" is always a 16-bit number. On a 16-bit machine, a word is represented as a single stack entry. On a 32-bit machine, a word is represented as the low 16 bits of a stack entry, with the upper 16 bits set to zero.

A "character" is always an 8-bit number. A character is represented as the low eight bits of a single stack entry, with the remaining upper bits set to zero.

## Address Incrementing Words

Changing all occurrences of `2+` to `4+` doesn't solve the problem, it just

sweeps it under a different rug. What we really need is a way to increment an address by the right number, regardless of what machine we're on. To do this, we define some names for the sizes of things.

**/N** (-- n) "per-n"

The number of bytes in a "normal" number, which is a single stack entry. **/N** is four on a 32-bit machine and two on a 16-bit machine.

**/L** (-- n) "per-l"

The number of bytes in a 32-bit "longword." **/L** is four on all machines.

**/W** (-- n) "per-w"

The number of bytes in a 16-bit "word." **/W** is two on all machines.

**/C** (-- n) "per-c"

The number of bytes in an 8-bit "character." **/C** is two on all machines.

The notation **/X** for "the number of bytes in an X," pronounced "per-x," follows the recommendations in Kim Harris's nomenclature guidelines ("Forth Coding Conventions," *Proceedings of the 1985 FORML Conference*).

You might think, since **/L** is always four, that the name "**/L**" is not needed. Ditto for **/W** and **/C**. However, the symbolic name **/L** clearly indicates that the code is dealing with the size of a longword, rather than the number four, which could be anything — perhaps the expected number of legs on a cow. Magic numbers make programs harder to understand and maintain!

Others have suggested the names **CELL** and **LSIZE** instead of **/N**; however, the name "normal" and the mnemonic "N" will be useful to us later.

What will we do with these constants? One obvious answer is to replace occurrences of **2+** with **/N**. Similarly, in cases where we want to step through an array of 16-bit words or 32-bit longwords, we might use **/W+** or **/L+**.

Another use is to calculate the number of bytes to **ALLOT** for some data structure or array. For instance, if we need space for 100 normal numbers, we could write **100 /N \* ALLOT** instead of **100 2\* ALLOT**.

A third use is to index into an array. Instead of writing the code in Figure One-a, for instance, we might use instead the definitions in Figure One-b. Notice that in all these cases, we have given up some efficiency! The word **2+** probably executes faster than the two words **/N +** and **2\*** almost certainly is faster than **/N \***. We will not tolerate such

# Fifth 2.5

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

inefficiency! Therefore, we define some more words, their existence amply justified by frequent use.

The functions in Figure Two are presently performed with 1+, 2+, and 4+, whose use does not work on all machines. Most occurrences of 2+ in existing Forth code can be replaced by NA1+ to make the code more transportable. The names stand for "normal-address-one-plus," etc., indicating that they increment an address to the next datum of a particular type.

Some machines do not directly address bytes. For instance, the Novix Forth chip is a word-addressed machine. Adding one to an address moves to the next 16-bit word, not to the next byte. For such machines, NA1+ is not equivalent to /N +. The real rule is that NA1+ should increment an address to point to the next item of a given type.

The words in Figures Three and Four find the address of the nth item in an array of items starting at addr. For instance, NA+ is equivalent to /N\* + on most

machines.

This may seem like a lot of words. It is a lot, but they are frequently used, which is the same justification used for words like 1+ and 2\*.

#### Explicit 32-bit Operators

One solution to the double-number problem on 32-bit machines is to make double numbers 64 bits. This is attractive because it is compatible with existing code that manipulates double numbers as pairs of stack entries. On the other hand, it is inefficient. 64-bit arithmetic is slower than 32-bit arithmetic on most machines. While many applications require more than 16 bits of precision, few require more than 32.

I believe the best long-term solution is to define a set of words that explicitly operates on 32-bit data, regardless of the machine's word size. The names of these words begin with the letter L, indicating that they operate on "long" operands. Their implementation is simple. On a 16-bit machine, they are the same as the

existing "D" words (e.g., D+) and the "2" words (e.g., 2DROP). On a 32-bit machine, they are the same as the regular single-number operators. The important point is that the "L" operators *always* operate on 32-bit longwords, regardless of machine size.

#### Long Arithmetic Operators

Some 32-bit arithmetic operators:

**L+** (L1 L2 -- L3) "l-plus"

Adds 32-bit longwords. On a 16-bit machine, L+ is the same as D+. On a 32-bit machine, L+ is the same as +.

**L-** (L1 L2 -- L3) "l-minus"

Subtracts 32-bit longwords. On a 16-bit machine, L- is the same as D-. On a 32-bit machine, L- is the same as -.

**L\*** (L1 L2 -- L3) "l-times"

Multiplies 32-bit longwords. On a 16-bit machine, L\* is the same as D\* (which is not included in the standard). On a 32-bit machine, L\* is the same as \*.

# Fifth 2.5

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

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

*Professional Version: \$150.00*

*Shareware Version, Disk and Manual: \$ 40.00*

*Shareware Version, Disk: \$ 10.00*

*System Source Code Available*

**CLICK Software**

**P.O. Box 10162**

**College Station, TX 77840**

**(409)-696-5432**

MS-DOS is a registered trademark of Microsoft Corp.

IBM is a registered trademark of International Business Machines Corp.

### L/ (L1 L2 -- L3) "1-divide"

Divides 32-bit longwords. On a 16-bit machine, L/ is the same as D/ (which is not included in the standard). On a 32-bit machine, L/ is the same as /.

I haven't mentioned all the operators that are needed, but the rest of them are named in the obvious way. For instance, L= compares two 32-bit longwords for equality.

### Stack Manipulations

The "2" stack operators, such as 2SWAP and 2DUP, are unsatisfactory for manipulating 32-bit longwords. Such operators were originally intended for manipulating pairs of numbers, which are distinctly different from 32-bit longwords. I propose a set of 32-bit stack manipulation operators whose names begin with (you guessed it) the letter L. Examples are LSWAP and LDUP.

Mixing 32-bit numbers and 16-bit numbers on the stack poses problems. In a 32-bit system, all stack entries are 32 bits, so this is not too bad. On a 16-bit system, both 32-bit and 16-bit numbers may need to coexist on the stack. Currently, this is handled in an ad hoc fashion, using operators like ROT to separately manipulate the pieces of the numbers. Programs that do this are not portable to 32-bit machines (here I assume that we have decided against using 64-bit numbers). What we need is a set of operators for manipulating mixed stacks. The needed operators mostly duplicate existing functions, so we really don't need new capability, just new names! The new names will clearly specify the sizes of the operands.

### LDUP (L -- LL) "1-dupe"

Duplicates a 32-bit longword. On a 16-bit machine, LDUP is equivalent to 2DUP. On a 32-bit machine, LDUP is equivalent to DUP.

### LSWAP (L1 L2 -- L2 L1) "1-swap"

Exchanges 32-bit longwords. On a 16-bit machine, LSWAP is equivalent to 2SWAP. On a 32-bit machine, LSWAP is equivalent to SWAP.

### LOVER (L1 L2 -- L1 L2 L1) "1-

```
CREATE MYARRAY 100 2* ALLOT
: FILLIT ( -- ) 100 0 DO I MYARRAY I 2* + ! LOOP ;
```

Figure 1a.

```
CREATE MYARRAY 100 /N * ALLOT
: FILLIT ( -- ) 100 0 DO I MYARRAY I /N * + ! LOOP ;
```

Figure 1b.

```
NA1+ ( addr -- addr+/n ) "n-a-one-plus"
LA1+ ( addr -- addr+/l ) "l-a-one-plus"
WA1+ ( addr -- addr+/w ) "w-a-one-plus"
CA1+ ( addr -- addr+/c ) "c-a-one-plus"
```

Figure 2. Words to increment an address by the appropriate amount.

```
/N* ( n -- n*/n ) "per-n-times"
/L* ( n -- n*/l ) "per-l-times"
/W* ( n -- n*/w ) "per-w-times"
/C* ( n -- n*/c ) "per-c-times"
```

Figure 3. Words to scale by different sizes.

```
NA+ ( addr n -- addr+n*/n ) "n-a-plus"
LA+ ( addr n -- addr+n*/l ) "l-a-plus"
WA+ ( addr n -- addr+n*/w ) "w-a-plus"
CA+ ( addr n -- addr+n*/c ) "c-a-plus"
```

Figure 4. Words to index into arrays.

over" Copies a 32-bit longword over a 32-bit longword. On a 16-bit machine, LOVER is equivalent to 2OVER. On a 32-bit machine, LOVER is equivalent to OVER.

### LDROP (L1 --) "1-drop"

Removes a 32-bit longword from the stack. On a 16-bit machine, LDROP is equivalent to 2DROP. On a 32-bit machine, LDROP is equivalent to DROP.

### LROT (L1 L2 L3 -- L2 L3 L1) "1-rote"

Rotates 32-bit longwords. On a 16-bit machine, LROT is equivalent to 2ROT. On a 32-bit machine, LROT is equivalent to DROT.

### LNSWAP (Ln -- nL) "1-n-swap"

Exchanges a 32-bit longword with a normal number. On a 16-bit machine, LNSWAP is equivalent to ROT ROT. On a 32-bit machine, LNSWAP is equivalent to SWAP.

### NLSWAP (nL -- Ln) "n-l-swap"

Exchanges a normal number with a 32-bit longword. On a 16-bit machine,

NLSWAP is equivalent to ROT. On a 32-bit machine, NLSWAP is equivalent to SWAP.

### LNOVER (Ln -- LnL) "1-n-over"

Copies a 32-bit longword over a normal number. On a 16-bit machine, LNOVER is equivalent to 2 PICK 2 PICK. On a 32-bit machine, LNOVER is equivalent to OVER.

### NLOVER (nL -- nLn) "n-l-over"

Copies a normal number over a 32-bit longword. On a 16-bit machine, NLOVER is equivalent to 2 PICK. On a 32-bit machine, NLOVER is equivalent to OVER.

### L>R (L --) "1-to-r"

Moves a 32-bit longword to the return stack.

### LR> (-- L) "1-r-from"

Moves a 32-bit longword from the return stack.

L>R and LR> are provided to help with more complicated stack manipulations

(Continued on page 25.)

# FIG MAIL ORDER FORM

## MEMBERSHIP IN THE FORTH INTEREST GROUP

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

The Forth Interest Group is a world-wide, non-profit, member-supported organization with over 4,000 members and 90 chapters. FIG membership includes a subscription to the bi-monthly publication, FORTH Dimensions. FIG also offers its members group health and life insurance, an on-line data base, a large selection of Forth literature and many other services. Cost is

\$30.00 per year for USA, Canada & Mexico; all other countries \$42.00 per year. The annual membership dues are based on the membership year, which runs from May 1 to April 30.

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

### ■ HOW TO USE THIS FORM

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

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

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

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

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

### ■ FORTH DIMENSIONS BACK VOLUMES

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

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

### ■ FORML CONFERENCE PROCEEDINGS

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

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

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

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

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

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

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

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

### ■ BOOKS ABOUT FORTH

200 — ALL ABOUT FORTH \$25/26/35 \_\_\_\_\_  
Glen B. Haydon

An annotated glossary for MVP Forth; a 79-Standard Forth.

216 — DESIGNING & PROGRAMMING PERSONAL EXPERT SYSTEMS \$19/20/29 \_\_\_\_\_

Carl Townsend and Dennis Feucht

Introductory explanation of AI-Expert System Concepts. Create your own expert system in Forth. Written in 83-Standard.

- 217 — F83 SOURCE \$20/21/30 \_\_\_\_\_  
 Henry Laxen & Michael Perry  
 A complete listing of F83 including source and shadow screens. Includes introduction on getting started.
- 218 — FOOTSTEPS IN AN EMPTY VALLEY (NC4000 Single Chip Forth Engine) \$25/26/35 \_\_\_\_\_  
 Dr. C. H. Ting  
 A thorough examination and explanation of the NC4000 Forth chip including the complete source to cmForth from Charles Moore.
- 219 — FORTH: A TEXT AND REFERENCE \$22/23/33 \_\_\_\_\_  
 Mahlon G. Kelly & Nicholas Spies  
 A textbook approach to Forth with comprehensive references to MMS-FORTH and the 79 and 83 Forth Standards.
- 220 — FORTH ENCYCLOPEDIA \$25/26/35 \_\_\_\_\_  
 Mitch Derick & Linda Baker  
 A detailed look at each fig-FORTH instruction.
- 225 — FORTH FUNDAMENTALS, V.1 \$16/17/20 \_\_\_\_\_  
 Kevin McCabe  
 A textbook approach to 79-Standard Forth
- 230 — FORTH FUNDAMENTALS, V.2 \$13/14/18 \_\_\_\_\_  
 Kevin McCabe  
 A glossary.
- 232 — FORTH NOTEBOOK \$25/26/35 \_\_\_\_\_  
 Dr. C. H. Ting  
 Good examples and applications. Great learning aid. PolyFORTH is the dialect used. Some conversion advice is included. Code is well documented.
- 233 — FORTH TOOLS \$22/23/32 \_\_\_\_\_  
 Gary Feierbach & Paul Thomas  
 The standard tools required to create and debug Forth-based applications.
- 235 — INSIDE F-83 \$25/26/35 \_\_\_\_\_  
 Dr. C. H. Ting  
 Invaluable for those using F-83.
- 240 — MASTERING FORTH \$18/19/22 \_\_\_\_\_  
 Anita Anderson & Martin Tracy  
 A step-by-step tutorial including each of the commands of the Forth-83 International Standard; with utilities, extensions and numerous examples.
- 245 — STARTING FORTH, 2nd Edition (soft cover) \$20/21/30 \_\_\_\_\_ **NEW**  
 Leo Brodie  
 In this new edition of Starting Forth, the most popular and complete introduction to Forth, syntax has been expanded to include the new Forth '83 Standard.
- 246 — STARTING FORTH (hard cover) \$20/21/30 \_\_\_\_\_  
 Leo Brodie
- 255 — THINKING FORTH (soft cover) \$16/17/20 \_\_\_\_\_  
 Leo Brodie  
 The sequel to "Starting Forth". An intermediate text on style and form.
- 265 — THREADED INTERPRETIVE LANGUAGES \$25/26/35 \_\_\_\_\_ **NEW**  
 R. G. Loelinger  
 Step-by-step development of a non-standard Z-80 Forth.
- 267 — TOOLBOOK OF FORTH \$23/25/35 \_\_\_\_\_  
 (Dr. Dobb's)  
 Edited by Marlin Ouverson  
 Expanded and revised versions of the best Forth articles collected in the pages of Dr. Dobb's Journal.
- 270 — UNDERSTANDING FORTH \$3.50/5/6 \_\_\_\_\_  
 Joseph Reymann  
 A brief introduction to Forth and overview of its structure.

### ■ ROCHESTER PROCEEDINGS

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

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

### ■ THE JOURNAL OF FORTH APPLICATION & RESEARCH

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

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

### ■ DR. DOBB'S JOURNAL

This magazine produces an annual special Forth issue which includes source-code listing for various Forth applications.

- 422 — DR. DOBB'S 9/82 \$5/6/7 \_\_\_\_\_
- 423 — DR. DOBB'S 9/83 \$5/6/7 \_\_\_\_\_
- 424 — DR. DOBB'S 9/84 \$5/6/7 \_\_\_\_\_
- 425 — DR. DOBB'S 10/85 \$5/6/7 \_\_\_\_\_

■ **HISTORICAL DOCUMENTS**

501 — KITT PEAK PRIMER \$25/27/35 \_\_\_\_\_

One of the first institutional books on Forth. Of historical interest.

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

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

503 — USING FORTH \$20/21/22 \_\_\_\_\_

FORTH, Inc.

■ **REFERENCE**

305 — FORTH 83-STANDARD \$15/16/18 \_\_\_\_\_

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

300 — FORTH 79-STANDARD \$15/16/18 \_\_\_\_\_

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

■ **REPRINTS**

420 — BYTE REPRINTS \$5/6/7 \_\_\_\_\_

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

■ **ASSEMBLY LANGUAGE SOURCE CODE LISTINGS**

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

514 — 6502/SEPT 80 \$15/16/18 \_\_\_\_\_

515 — 6800/MAY 79 \$15/16/18 \_\_\_\_\_

516 — 6809/JUNE 80 \$15/16/18 \_\_\_\_\_

517 — 8080/SEPT 79 \$15/16/18 \_\_\_\_\_

518 — 8086/88/MARCH 81 \$15/16/18 \_\_\_\_\_

519 — 9900/MARCH 81 \$15/16/18 \_\_\_\_\_

521 — APPLE II/AUG 81 \$15/16/18 \_\_\_\_\_

523 — IBM-PC/MARCH 84 \$15/16/18 \_\_\_\_\_

526 — PDP-11/JAN 80 \$15/16/18 \_\_\_\_\_

527 — VAX/OCT 82 \$15/16/18 \_\_\_\_\_

528 — Z80/SEPT 82 \$15/16/18 \_\_\_\_\_

■ **MISCELLANEOUS**

601 — T-SHIRT SIZE \_\_\_\_\_

"May the Forth Be With You"

Small, Medium, Large and Extra-Large

White design on a dark blue shirt. \$10/11/12 \_\_\_\_\_

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

616 — HANDY REFERENCE CARD FREE \_\_\_\_\_

683 — FORTH-83 HANDY REFERENCE CARD FREE \_\_\_\_\_

■ **FORTH MODEL LIBRARY**

The model applications disks described below are new additions to the Forth Interest Group's library. These disks are the first releases of new, professionally developed Forth applications disks. Prepared on 5 1/4" disks, they are IBM MS-DOS 2.0 and up compatible. The disks are compatible with Forth-83 systems currently available from several Forth vendors. Macintosh 3 1/2" disks are available for MasterFORTH systems only.

Forth-83 Compatibility IBM MS-DOS

Laxen/Perry F83

LMI PC/FORTH 3.0

MasterFORTH 1.0

TaskFORTH 1.0

PolyFORTH

**Forth-83 Compatibility Macintosh MasterFORTH**

**Ordering Information**

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

by Martin J. Tracy

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

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

by Craig A. Lindley

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

703 — AUTOMATIC STRUCTURE CHARTS V.3 . \$40/43/45 \_\_\_\_\_

by Kim R. Harris

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

704 — A SIMPLE INFERENCE ENGINE V.4 \$40/43/45 \_\_\_\_\_ **NEW**

by Martin J. Tracy

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

706 — THE MATH BOX V.6 \$40/43/45 \_\_\_\_\_ **NEW**

by Nathaniel Grossman

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

Please specify disk size when ordering \_\_\_\_\_

■ **NC4000SERIES**

801 — MORE ON NC4000, VOLUME 1 \$10/11/14 \_\_\_\_\_ **NEW**

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

802 — MORE ON NC4000, VOLUME 2 \$15/16/18 \_\_\_\_\_ **NEW**

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

803 — MORE ON NC4000, VOLUME 3 \$15/16/18 \_\_\_\_\_ **NEW**

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

804 — MORE ON NC4000, VOLUME 4 \$15/16/18 \_\_\_\_\_ **NEW**


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

# FORTH INTEREST GROUP

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

Name \_\_\_\_\_  
 Member Number \_\_\_\_\_  
 Company \_\_\_\_\_  
 Address \_\_\_\_\_  
 City \_\_\_\_\_  
 State/Prov. \_\_\_\_\_ ZIP \_\_\_\_\_  
 Country \_\_\_\_\_  
 Phone \_\_\_\_\_

OFFICE USE ONLY			
By _____	Date _____	Type _____	
Shipped by _____	Date _____		
UPS Wt. _____	Amt. _____		
TNT Wt. _____	Amt. _____		
USPS Wt. _____	Amt. _____		
BO Date _____	By _____		
Wt. _____	Amt. _____		

ITEM #	TITLE	AUTHOR	QTY	UNIT PRICE	TOTAL
109	<b>MEMBERSHIP</b>				<b>SEE BELOW</b>

CHECK ENCLOSED (Payable to: Forth Interest Group)

VISA     M/C

Card # \_\_\_\_\_

Expiration Date \_\_\_\_\_

Signature \_\_\_\_\_

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

<b>SUB-TOTAL</b>	
<b>ORDERS OF \$50.00 OR MORE RECEIVE A 10% DISCOUNT</b>	
<b>SUB-TOTAL</b>	
<b>CA. RESIDENTS ADD SALES TAX</b>	
<b>HANDLING FEE</b>	<b>\$2.00</b>
<b>MEMBERSHIP</b>	
<input type="checkbox"/> NEW <input type="checkbox"/> RENEWAL \$30/42	

## PAYMENT MUST ACCOMPANY ALL ORDERS

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

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

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

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

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

**SALES TAX**  
 Deliveries to Alameda,  
 Contra Costa, San Mateo,  
 Los Angeles, Santa Cruz  
 and San Francisco Counties,  
 add 6 1/2%. Santa Clara  
 County, add 7%; other  
 California counties, add 6%.



(Continued from page 20.)

involving mixed stacks. (However, it is usually preferable to try to avoid complex stack gymnastics, instead.)

#### Accessing Memory

We need some words for accessing memory items of various sizes. We already have `C@`, `C!`, `2@`, `2!`, `@`, and `!`. We also need some words to access exactly 16 bits and exactly 32 bits, so we add these words:

`W@ (adr -- 16b) "w-fetch"`

Fetches the 16-bit word at `adr`. On a 32-bit machine, the result is padded with zero to form a 32-bit normal number on the stack.

`<W@ (adr -- 16b) "signed-w-fetch"`

Fetches the 16-bit word at `adr`. On a 32-bit machine, the result is sign-extended to form a signed, 32-bit normal number on the stack.

`W! (16b adr --) "w-store"`

Stores the 16-bit word at `adr`. On a 32-bit machine, the number "16b" is represented on the stack as the lower half of a 32-bit normal number, and only the lower 16 bits are stored at `adr`.

`L@ (adr -- L) "l-fetch"`

Fetches the 32-bit longword at `adr`. On a 16-bit machine, the result is left on the stack as two 16-bit numbers, with the most-significant half on the top of the stack.

`L! (L adr --) "l-store"`

Stores the 32-bit longword "L" at `adr`. On a 16-bit machine, the longword "L" is represented on the stack as two 16-bit numbers, with the most-significant half on the top of the stack.

#### Type Conversion

Some words for converting to and from 32-bit longwords:

`N->L (n -- L) "n-to-l"`

Converts a normal number to an unsigned 32-bit longword. On a 16-bit machine, `N->L` is equivalent to `0`. Does nothing on a 32-bit machine.

`S->L (n -- L) "signed-to-long"`

Converts a normal number to a signed 32-

bit longword. On a 16-bit machine, `S->L` is equivalent to `S->D`. Does nothing on a 32-bit machine.

`L->N (L -- n) "l-to-n"`

Converts a 32-bit longword to a normal number. On a 16-bit machine, `L->N` is equivalent to `DROP`. Does nothing on a 32-bit machine.

#### Important Note

It is neither necessary nor desirable to use the "L" operators all the time on a 32-bit system. Most of the time you don't really care whether a number is 16 bits or 32 bits. So you just use the normal operators like `+`, `-`, `DUP`, etc. The right time to use the "L" operators is when:

1. You require more than 16 bits, and
2. You want your code to run on both 16-bit and 32-bit machines.

In particular, I use the "L" operators when converting "double numbers" from 16-bit machines to 32-bit machines. Also remember that "double numbers" on 16-bit machines are not always the same as 32-bit longwords. Sometimes the "double number" operators are used to manipulate *pairs* of single numbers. Such uses must remain unchanged. A pair of numbers is still a pair of numbers, even on a 32-bit machine.

#### The Name's the Thing

The underlying problem is that Forth often uses the same name for different purposes. Examples: `2+` is used to add the number two, and to increment an address to the next word; `2DROP` is used to remove two single numbers from the stack, and to remove a 32-bit number from the stack. This happens to have worked in the past, because Forth just assumed that every machine in the world is a 16-bit, byte-addressed machine. (In fact, it didn't work on the Data General Nova, which is word addressed!) Different conceptual functions should have different names, even if the functions happen to have identical implementations on a particular machine.

That is why I am proposing new names without any new functions!



#### FIG-FORTH for the Compaq,

IBM-PC, and compatibles. \$35

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

Full-screen editor uses 16 x 64 format.

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

Source included for the editor and other utilities.

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

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

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

#### Async Line Monitor

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

#### A Metacompiler on a

host PC, produces a PROM

for a target 6303/6803

Includes source for 6303

FIG-Forth. Application code

can be Metacompiled with

Forth to produce a target

application PROM \$280

#### FIG-Forth in a 2764 PROM

for the 6303 as produced by

the above Metacompiler.

Includes a 6 screen RAM-Disk

for stand-alone operation. \$45

#### An all CMOS processor

board utilizing the 6303.

Size: 3.93 x 6.75 inches.

Uses 11-25 volts at 12ma,

plus current required for

options. \$210 - \$280

Up to 24kb memory: 2 kb to

16kb RAM, 8k PROM contains

Forth. Battery backup of RAM

with off board battery.

Serial port and up to 40 pins of

parallel I/O.

Processor buss available at

optional header to allow expanded

capability via user provided

interface board.

#### Micro Computer Applications Ltd

8 Newfield Lane

Newtown, CT 06470

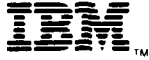
203-426-6164

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

# PORTABLE POWER WITH MasterFORTH



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



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

Forth is interactive - you have immediate feedback as you program, every step of the way. Forth is



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

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

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

(213) 821-4340



**MICROMOTION**

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

## Yes, But ...

If we make all these new names for functions we already have, like

```
: LSWAP ( L1 L2 -- L2 L1 )
2SWAP ;
```

doesn't the code run slower? There are two answers:

1. It's not much slower.
2. There's a way to get the speed back.

To speed it up, see the article on synonyms (Yngve, Victor H. "Synonyms," *Forth Dimensions* VII/3). A synonym provides a new name for an existing word, with no run-time penalty. Basically, a synonym is an extra name for an existing word. When the compiler sees the new name, it compiles the compilation address of the existing word.

## Tips for Converting Programs

When I convert a 16-bit program to a 32-bit machine, the first thing I do is search for all occurrences of the character 2. I look for the number two and for words that start with 2, such as 2DROP. Many of these have to change. Usually, words like 2+ change to NA1+, and 2\* to /N\*, etc. For words like 2DROP, you have to decide whether it is being used to drop a pair of normal numbers (in which case you leave it alone), or a 32-bit number (in which case it changes to LDROP).

The next thing I look for is words like D+, D-, etc., changing them to L+,L-, etc.

Sometimes the word 0 is used to convert an unsigned normal number to a longword. This has to change to N->L.

Finally, I look for cases where stack manipulations like ROT are used to swap mixed 32-bit and 16-bit numbers. These change to words like NLSWAP. These are not as hard to find as you might guess, because they usually occur just before a 32-bit operator, so you should keep an eye out for them while looking for 2, D+,0, etc.

Since each of the words proposed here can be implemented so easily, it is easy to add them to the program as you need them. I usually group their definitions together at the beginning of the source

code, to make them easier to find.

Sometimes you may find you need a mixed 16-bit/32-bit operator not mentioned here. I recommend that you pick a name starting with either "NL" (if the longword is on the top of the stack) or "LN" (if the normal number is on top).

## Further Reading

This wordset was originally proposed in two earlier papers.

1. Bradley, Mitch and Sebok, Bill. "Compatible Forth on a 32-bit Machine," *The Journal of Forth Application and Research*, Vol. 2 No. 4, 1984.

2. Bradley, Mitch and Sebok, Bill. "Extended Addressing Wordset," Working Group Report, *Proceedings of the 1984 Rochester Forth Conference*.

These papers describe some other words not mentioned here, including names for words to perform extended addressing on 16-bit machines. Vol. 2 No. 4 of *The Journal of Forth Application and Research* contains several papers describing other aspects of 32-bit machines.

*Mitch Bradley is the owner of Bradley Forthware, vendor of the 32-bit "Forth-macs," a very complete Forth-83 environment for the Atari ST.*

# STARTING FORTH INC.

AN INTERVIEW WITH ELIZABETH RATHER

**E**lizabeth Rather is the president of FORTH, Inc. and was as close to Forth's beginnings as anyone except its creator. At a recent convention, Michael Ham interviewed Ms. Rather at length about the early years of Forth and the history of the first company to deal in Forth systems and applications. She spoke frankly of product successes, lean times in the early days, and public perception.

**Michael Ham:** I'd like to get a sense of the history of FORTH, Inc., and then a sense of where it is today and what its plans are for the next thrust in Forth. I feel there have been many changes in terms of product packages and building up in another, new direction. So let's start with the history.

**Elizabeth Rather:** I want to go back to the actual dawn of time, which was a day in 1971. I was working at the University of Arizona, in the registrar's office, processing student records. My then husband was an astronomer at the National Radio Astronomy Observatory, also in Tucson. I walked into the lab one day and there was a rack of electronic equipment in the middle of the lab — which was not unusual — and there was a guy sitting in front of it. I asked, "What's that?" and he said, "That's a computer."

I was absolutely amazed, because I knew what a computer was: a computer occupied an entire room with a raised floor and had thousands of acolytes around it. This little box in the middle of the room with a guy sitting in front of it was something I'd never dealt with before. I came to understand that this little

computer was going to be controlling the telescope and doing data acquisition and analysis, and all kinds of neat stuff. I asked, "How are you programming it?" He said, "Well, it's a language called Forth." I asked, "What is that?" And the rest is history.

**MH:** You had programmed before?

**ER:** At that time, I had been programming for about ten years in Fortran, assembly language, and a little bit of Cobol (I'm almost embarrassed to admit). I'd gotten very interested in some of the avant garde things that were going on. I'd been playing a little bit, just the previous week, as a matter of fact, with APL. I'd really fallen in love with APL, but it was doomed to be a very short love affair because I fell in love with Forth shortly afterwards.

**MH:** It shows you had a weakness for weird languages, at least.

**ER:** Right, and I realized it was really the interactiveness of APL that appealed to me, and not anything particular about it as a language. Going at a program interactively is something I still think the world doesn't value nearly enough. Even the languages people are using today, such as C, are not interactive, and it is hard to realize how incredibly powerful interactivity is.

**MH:** Right. I'll put in one story of mine: people don't understand that when you say it is interactive, it means you can do this back-and-forth with the program. They say, "Yeah, but so what?" I

remember the first time I heard about a text editor. It was Wylbur, and I asked, "So, what does it do?" My friend explained to me, "You can correct words, and search and find them, and move lines around..." And I said, "That's it? So what?" I didn't understand at the time, but now I can't write without a word processor.

When you found Forth, did you start programming in it right then?

**ER:** No, I didn't. I'd always been interested in the concept of computers making things wiggle, which was actually a fairly new concept at that time. I was also pretty bored with the registrar's office. Chuck was living at that time in Charlottesville, Virginia, and he would come out for a couple of months at a time, reprogram everything in sight, and then go back to Charlottesville, leaving chaos in his wake. Ned Conklin, who ran the Tucson division of the observatory, was a very good friend of ours, and he would come over for dinner, and wring his hands and cry a lot about this guy who wrote these beautiful programs that were always, chronically, ninety percent done.

I took advantage of the situation and volunteered to take on, as a moonlighting operation, the task of understanding the programs and providing some local support, talking on the phone with Chuck as need be, and solving problems as they arose. We originally thought it was going to be ten hours a week, and I found myself working for them over forty hours a week in addition to my regular job. We figured it was time to make some adjustments.

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

## The MMSFORTH System. Compare.

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

# MMSFORTH

## A World of Difference!

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

- **Personal License (required):**  
MMSFORTH V2.4 System Disk . . . \$175.00  
(TRS-80 Model 1 requires lowercase, DDEN, 1 40-track drive.)
- **Personal License (additional modules):**  
PORTCOMM communications module . . . \$ 49.00  
UTILITIES . . . . . \$ 49.00  
GAMES . . . . . \$ 39.00  
EXPERT-3 expert system . . . . . \$ 99.00  
DATAHANDLER . . . . . \$ 59.00  
DATAHANDLER-PLUS (PC only, 128K req.) . . . \$ 99.00  
FORTHWRITE word processor . . . . . \$ 59.00
- **Corporate Site License Extensions** . . . . . from \$1,200
- **Bulk Distribution** . . . . . from \$250/95 units.
- **Some recommended Forth books:**  
FORTH A TEXT & REF. (best text) . . . . . \$ 15.00  
TEACHING FORTH (best on techniques) . . . \$ 15.00  
STARTING FORTH (popular text) . . . . . \$ 9.95

Shipping/handling & tax extra. No returns on software.  
Ask your dealer to show you the world of MMSFORTH, or request our free brochure.

**SELLER MICROCOMPUTER SERVICES**  
81 Lake Shore Road, Rindge, MA 01128  
(617) 650-5132

MH: That must have been like trying to learn Chinese by trying to read a newspaper with the aid of a dictionary. That's a hard way to get into Forth.

ER: Well, contrary to popular opinion, Chuck does write. He is capable of it. In fact, he writes extremely well, and at that time he had written three volumes of documentation of Forth. In fact, he wrote a book — which is a wonderful book — called *Programming a Problem-Oriented Language*. I still have a copy in my office, and I keep cherishing the hope that someday I can edit and publish it, because it was about Forth at a time slightly before I met Chuck. It was a previous incarnation of Forth, but very recognizable. The book tells why it was like it was, how he was making his decisions. It was written in a style that is absolutely just like hearing Chuck talk. It's wonderful to read from that point of view, and it's very interesting as a historical document of how an extraordinarily bright person goes about thinking of a new language. I don't think that's ever been documented before. It really is almost a stream of consciousness account of how that took place, and it was wonderful. But it didn't really explain a whole lot about how the current system worked, and it certainly looked funny. I really still sympathize with people who have never seen Forth before, who look at it and say, "Boy, is that ever a funny-looking language."

I can remember how that felt. The other documentation he wrote tended to describe individual words. You had to know what the words were in order to look them up. It really was not as helpful as it might have been, so one of the very early things I did at the observatory was write a user's manual on Forth per se. I think that was the first user's manual on Forth, and it was the first step on a road of moral decay that's led to the current polyFORTH reference manual, which I've just spent the last six months of my life re-editing.

MH: And repackaging the entire thing in a handsome way.

ER: I added it up one time. FORTH,

Inc. has spent over one hundred thousand dollars on that reference manual, in terms of manpower costs.

MH: It's a cumulative distillation of experience, as I look at it. It's what you need to know, really. I've noticed programmers often don't want to take you into their confidence. They tell you the minimal stuff that's actually correct about the program, but they won't say, "Look, here's what you need to know." But the reference manual tells you what you need to know.

When did you run into Chuck in person — did he come out on one of his visits?

ER: Yes, it was on one of his visits. I encountered him sitting in the lab, and subsequently we had Chuck and Min over to dinner at our house many times. We became very good friends. After I got a job working full-time at the observatory, we went through a major upgrade of the observatory software and hardware and developed a system that has survived at the observatory—evolving, of course, but without really significant change — for over ten years, which in that kind of research environment is an absolutely incredible record of longevity. The reason is, when we installed it, it was so far in advance of the state of the art that it stayed an advanced system for a very long time. When we were near to finishing that project, I recall very clearly, we were sitting around our patio one summer evening in Tucson talking about what the future was going to hold for computers and mini-computers and programming, and what was going to happen to Forth, and so on. We had given away a number of systems to astronomers who had come there — we were a service organization, and astronomers came from all over the world to use that facility. A number of them were so impressed with the software that they took copies home and tried to adapt it for use. It had become apparent that Forth was as powerful for other people as it was for us. So this particular evening, we said, "You know, why don't we form a company and do something? There's got to be some money in it somewhere."

So FORTH, Inc. was born that evening. It was in the summer of 1973. Chuck and his wife Min, myself and my husband John, and Ned Conklin, who was our mutual boss, went to see a lawyer the following week and incorporated FORTH, Inc.

**MH:** Was Forth primarily on one machine at that time?

**ER:** No, it was on a number of machines. It was on a PDP-11, which was the machine we upgraded. The very first was a Honeywell minicomputer; and it had migrated into the observatory on Varian 620s and Hewlett-Packard 2116s, and the Nova was across the street at Steward Observatory. So there were a number of systems.

**MH:** So Forth from its very beginning started this heritage of polyFORTH covering a broad range of machines and processors...

**ER:** Yes, that was true, and it's also true that the very first Forth systems were all multi-tasking, multi-user, standalone operating systems. Multi-tasking is absolutely not anything new or recent. It's been a mainstream of Forth development on every system that I have ever had personal experience with.

**MH:** Were Forth applications at the observatory strictly for telescope control and scientific computation? What range of applications was it being used on, and where did you see its use?

**ER:** Forth controlled the telescope, and it did high-speed data acquisition. We were recording 500 samples every 10 milliseconds, I believe, which is ticking along pretty well. We also had a very early Tektronix graphics terminal that we had a nice graphics package on, including some 3-D graphics capability and a lot of mathematical analysis. The object of the game was to send an astronomer home with his data mostly reduced on the spot; it was very important to allow him to reduce his data on the spot because he could adjust his observing program depending on what he was seeing or not seeing, as the case may be.

**MH:** Let's turn to your background. Your studies were not in computer science, as I recall.

**ER:** No, I have both an undergraduate and a graduate degree in medieval history. It's a lot of fun, but there's no money in it.

**MH:** Well, it does lead to a career in Forth. You began in computers just...

**ER:** I actually had studied physics as an undergraduate. I didn't complete a degree in physics, but I had studied it. I was married to an astronomer, and all our friends were scientists. I felt comfortable in the scientific community. I went to work in the days when any woman applying for a job was required to take a typing test. Regardless of what you said you were interested in, if you were a female and went into the personnel department, you took a typing test. It turns out, of course, that typing is one of the primary skills for a Forth programmer, so that was not inappropriate.

My first job was handling data processing for a physics research group at the Oak Ridge National Laboratory. They had an extremely primitive computer and some very primitive mainframes — I'm almost embarrassed to tell you when this was. I was working with the computers, and I convinced my boss, Jack Harvey, that I should be allowed to take a two-week Fortran course so I could handle the programs a little bit better. I did that and wrote a couple of very simple Fortran programs. Not long after that we moved to California. I went into the University of California's personnel office, and said, "I want a job." They asked, "What do you do?" and I said, "I'm a programmer."

They sent me for an interview with the astronomy department. I really wanted that job. The professor there had a group working on a NASA contract doing analysis of comet and satellite orbits, and he said, "We do all of our programming in assembly language on a 7094." I said, "Well, I have never actually written assembly language for the 7094." And he said, "Don't worry, we'll teach you." And they did.

**MH:** When you put your company together, did you start in Tucson?

**ER:** We never did any business as FORTH, Inc. in Arizona. We moved to California in late 1973, reincorporated the company there, and opened for business about then. I sent a press release to *Computerworld*. I had never written a press release before and didn't even know what one looked like. But I wrote one and sent it to several of the computer magazines. *Computerworld* ran it on the front page in about November of 1973. By December and January, the letters were flooding into our new home in Manhattan Beach, California. We got about 350 letters, and Chuck and Min moved to California about then. I had an old typewriter, and Chuck and I sat in the back room of my house, taking turns writing personal answers to all 350 letters. Out of that we started getting a few jobs, doing custom programming for people.

**MH:** Your first sale was to a man in New Jersey?

**ER:** Yes, Art Gravina of Cybek. He had been working with a very large BASIC program for business data processing. He had been the architect of the program, written in something called Extended BASIC. Art's dream was to do larger applications than this Extended BASIC could handle. He called us on the phone and we had several conversations. He said, "If what you say is real, we've got a tiger by the tail here." He came out for one weekend, Chuck and I flew up to northern California, and we all spent the weekend together at the offices of Systems Industries in Sunnyvale. They made a computer available to us starting at five o'clock Friday afternoon.

By Monday morning, we had ported Forth onto that computer from a paper tape we had taken with us from the observatory. We had to develop a new disk controller for it on the spot, hand coding in with the paper tape — it was unreal. Having gotten that done, we implemented all the rudiments of a database system, and wrote an application that would accept data, display data, write reports, and so on. We had all that done

by Monday morning. We hadn't had a whole lot of sleep, I must admit, but we had it all working, with several terminals up and the whole scene.

**MH:** It was a multi-user system?

**ER:** Yes, everything we've ever done has been multi-user. Art went back to New Jersey, quit his job the next day and founded a company. He has done extremely well since then, and has continued with large database applications, most recently in the area of hospital management. His company was recently bought by a division of McDonnell Douglas.

There was another hospital system written in polyFORTH that Honeywell is marketing. Bob Barnett, our second or third customer, wrote it. He's in Iowa, and he would go away for a couple of years, developing his applications, then call us back in, then go away again on his own.

**MH:** How big is FORTH, Inc. today, and what are your major projects at this time?

**ER:** We're back in Manhattan Beach after six years in Hermosa Beach. We have about 25 employees, most of them full-time, a few of them consultants. Our business is about two-thirds custom programming and one-third sales and support of polyFORTH. The custom programming that we're doing is primarily in areas such as automated manufacturing, robotics, process control, that sort of thing.

**MH:** Private industry more than government?

**ER:** We have done some government work, but the vast majority of our effort has been private industry. We have felt more comfortable there. The greatest part of our business is products.

**MH:** Are your clients in California or all over the country?

**ER:** They are all over the country. We have worked for aerospace companies,

major manufacturing companies, major engineering firms. From the beginning, we have focused mostly on large companies, although a number of our very successful customers have been start-up companies using polyFORTH to program a clever widget they have designed.

One of the most distributed polyFORTH applications is an automobile engine analyzer that was developed by a company called Allen Test Products in Kalamazoo, Michigan. It is sort of an expert system. It was originally written before expert systems were popular and it lacks some of the formalization that people expect of expert systems, but it really was a rule-based system, with definite rules in it. It takes an automobile mechanic through a series of tests and modifies the tests that will be performed based on the results and on what it is learning. It will pursue various avenues and, at the end, it will print out a report of what is wrong with the car and what it recommends should be done. This was picked up by Firestone and is in all the Firestone Master Care Centers across the United States. It's called the Master Mind of Firestone, and it's also sold privately as something called a Smart-Scope. It's gone all over the world. The operator-interaction screens have been translated into at least five languages.

**MH:** Do you ever wish it said, "Written in polyFORTH?"

**ER:** You bet I do.

**MH:** Were there any black moments in Forth's early history?

**ER:** It was a real hand-to-mouth existence for a while. We knew very little about marketing and very little about things like advertising. It is still true that people are reluctant to base major products on a programming language they've never heard of, and not nearly enough of them have heard about Forth. It was very hard. I was the marketing department and Chuck was the programmer. I spent days on the phone calling up people saying, "Can we do some work for you, can we talk?" And we

had days when we didn't know where the next nickel was coming from. In more recent times, there have been a couple of major downturns. But we've managed to weather all that. I went back to school and got an MBA, and that has helped.

**MH:** What were some of your big hits?

**ER:** There have been a large number of those. Wonderful moments early on with Art, when we gave demonstrations where we had done something really spectacular in no time at all. We replaced a program that was running on a 370, and the Forth version ran about four times as fast and did some things that the 370 hadn't been able to do at all. That was a lot of fun.

We've been the first serious software on a number of processors. Honeywell, in the late 70s, came out with a new minicomputer called the Level 6, and polyFORTH was the first operating system of any kind on it. It was the first high-level programming language on the Level 6. In order to achieve that, we worked for several weeks at Honeywell's facility in Massachusetts. We had access to the computer from five at night until five in the morning. We did that for several weeks. At the end of that time, we had polyFORTH up on it, and Honeywell couldn't care less. They were still busy designing their operating system.

We were the first high-level language to run on the 8086, by nearly two years. We were running on the 8086 when there were still bugs in the part. For a long time, the only thing Intel offered that ran on the 8086 was an assembler. Everything else was cross-assembled.

**MH:** I heard once that Radio Shack was thinking of putting Forth into their first personal computer instead of BASIC. Is that true?

**ER:** We had some very serious conversations with Radio Shack. I don't think they were looking at it as an alternative to BASIC, but they were looking at carrying it. They decided, after an intensive series of conversations, that there was not a market for it. Dick Miller certainly proved that wrong.

**MH:** Based on what people order from FORTH, Inc., what would you say is the most popular microprocessor today? Where is the excitement?

**ER:** It is completely dominated by the IBM. In our business, we also see a great deal of work on the 68000. It is used very heavily in the kinds of factory automation applications we work on.

**MH:** What other things do you think are significant in the development of FORTH, Inc., or Forth; other moves in our history that we haven't really touched on as you look back at watershed events and critical decisions.

**ER:** Goodness. I think probably one of the big watersheds was when we decided to go after microprocessors in 1977. That was when microprocessors were newborn. In early 1977, we implemented a very primitive version of Forth on the RCA 1802, which was a brand-new chip at that time. I gave a paper on it, jointly with some people at RCA, at Electro '77 in Boston. Seeing the interest in that and seeing what was clearly taking shape around 8080s and around Intel — which all had just been introduced at that time — we decided that was the wave of the future and we were going to go after it at FORTH, Inc.

On the basis of that, we brought out our first standard mail-order product, which was called microFORTH. Prior to that, all our systems were custom written and personally installed on minicomputers. With the advent of the microprocessor, we invested from our very slender capital — it was a very big risk for us at the time — and bought an Intel blue box development system in the MDS-800 then, and we invested a considerable amount of effort in bringing out our first standard mail-order product of microFORTH. We then implemented that on several microprocessors. microFORTH was a moderately successful product for us although, in retrospect, technically it had an awful lot of shortcomings. When we upgraded to polyFORTH less than two years later, it was a tremendous improvement. What I think was most significant about that was that we marketed it by selling, for as I

recall about \$15, a book called *microFORTH Primer*. We sold several thousand of those.

It was the people who picked up the *microFORTH Primer* but were unable to afford to buy microFORTH who formed the Forth Interest Group and all that activity. So that certainly has to be regarded as a watershed. It was a watershed for FORTH, Inc. because it first got us into the standard product business and standard product support. Without that, I don't think we would have survived or thrived as we have as an organization. But it also had the side effect of stimulating the formation of the Forth Interest Group and that whole community.

**MH:** Is your product line now, I imagine, primarily microcomputers? Do you have much mainframe stuff at all?

**ER:** We have never had any real mainframe software in the sense of IBMs and the like. We do work on some moderately large computers. We have a version of polyFORTH that runs under VMS on VAXes, and we have a thirty-two-bit system that runs on fairly good-sized 68000-based systems.

**MH:** If you had it to do over, what would you do differently?

**ER:** There are some interesting things I can say on that score. I think the major thing I would do differently is to take a much more serious interest in public relations in general. We've sponsored a number of magazine articles over the years, but we have never really had anything like the kind of intensive, continued, sustained public relations effort that I think we should have done and that, I think, is required to make Forth well known. Some of the spectacular things we've done, such as being the first high-level language, the first real operating system on some of these processors, the world doesn't know about.

The world, even today, does not know about some of the very, very successful Forth applications — not just polyFORTH, but Forth applications in general. People, to a great extent, still think that Forth is sort of a freak, a

hobbyist/hacker toy, and that serious people don't program in it. That is quite patently not the case. We at FORTH, Inc. have not done as good a job as we might have done, and the Forth community in general has not done as good a job as it should have done, at telling the story of just how successful companies have been using Forth.

**MH:** Have you seen polyFORTH adopted in universities?

**ER:** Oh, yes. We have a fairly aggressive university discount policy and have for a number of years, and some of them have been very successful with it. There is an excellent group at Stanford, a long-standing and active group at MIT, and literally hundreds of other universities from around the country that have picked it up.

**MH:** And that contributes to its use...

**ER:** It does, but not nearly enough. What we see is that in the universities this is happening in the engineering departments, where they really are not interested in Forth per se, or any computer language per se. They are interested in getting results out of experiments. Of course, Forth has always appealed to people who are focused on getting results. We don't see it penetrating the computer science departments because they believe their major *raison d'être* is to train conventional programmers to use conventional programming tools. And polyFORTH, or Forth in general, is not perceived in that way.

*Michael Ham currently works in large-scale data processing, and expects to have a Forth program published shortly for the IBM microcomputer. His work has appeared frequently in these pages and in other respected magazines.*

# RUN-TIME STACK ERROR CHECKING

CHARLES SHATTUCK-ROSEVILLE, CALIFORNIA

One of the biggest problems when trying to learn Forth, especially for naive beginners, is stack errors that crash the whole system. In a classroom environment, where students have only an hour or so to use a computer, the time spent rebooting can be significant. Add to that the frustration of not knowing why the system crashes, and you can get students who aren't very excited about Forth.

Laxen and Perry address part of the problem in their F83 system with the word `?ENOUGH`. This word checks to see that enough items are on the stack before execution, preventing accidents like listing screen zero, which always messes up the eighty-column card in my Apple IIe. Otherwise, the main problem seems to be stack overflow errors, which cause the Apple to hang irretrievably. To solve this problem for beginners, we need a way of checking the stack for correct output at the end of a word as well. It is considered good style to include a stack diagram with each word definition, so why not use these to automate stack error checking?

The word that does the job is `(S`. The beginning programmer can include a stack diagram after the name of a new word, and have the computer use it to figure out the number of inputs and outputs the word requires. At run time, if the expected number of values differs from the actual number, an error message is displayed. Stack errors are caught before they do too much damage, and the programmer has some idea of where and why the error occurred. The immediate feedback is helpful to beginning Forth programmers. The teacher just has to be sure the student

```
Scr# 32.....
0 \ Run-time stack error-checking                24Jun86 CWS
1
2 VARIABLE CHECKING CHECKING OFF
3 VARIABLE INPUTS VARIABLE OUTPUTS VARIABLE PARAMETERS
4
5 : ?INPUTS ( n1 n2 -- n3 ) OVER DEPTH 3 - >
6   ABORT" Not enough input parameters" - DEPTH 1- SWAP - ;
7 : ?OUTPUTS ( n ) DEPTH 1- = NOT
8   ABORT" Wrong number of output parameters" ;
9
10 : IN/OUT ( n1 n2 )
11   COMPILE ?INPUTS COMPILE >R CHECKING ON ; IMMEDIATE
12 : ; CHECKING @ IF COMPILE R) COMPILE ?OUTPUTS THEN
13   [COMPILED] ; CHECKING OFF ; IMMEDIATE
14
15

Scr# 33.....
0 \ Run-time stack error-checking                24Jun86 CWS
1
2 : PARAMETER? ( n -- ? )
3   DUP ASCII ? = IF 1 PARAMETERS @ +! TRUE ELSE
4   DUP ASCII n = IF 1 PARAMETERS @ +! TRUE ELSE
5   DUP ASCII d = IF 2 PARAMETERS @ +! TRUE ELSE
6   DUP ASCII - = IF OUTPUTS PARAMETERS ! TRUE ELSE
7   DUP ASCII ) = IF ( leave ) FALSE ELSE
8   TRUE ABORT" Stack picture error" THEN THEN THEN THEN THEN
9   SWAP DROP ;
10
11 : (S INPUTS OFF OUTPUTS OFF ?INPUTS PARAMETERS !
12   BEGIN BL WORD 1+ C@ PARAMETER? NOT UNTIL
13   INPUTS @ [COMPILED] LITERAL OUTPUTS @ [COMPILED] LITERAL
14   [COMPILED] IN/OUT ; IMMEDIATE
15

Scr# 34.....
0 \ Run-time stack error-checking, example       25Jun86 CWS
1
2 : ADD (S n1 n2 -- n3 ) + ;
3 \ Works fine if at least two numbers are on the stack at entry.
4
5 : BAD (S n1 n2 -- ) + ;
6 \ Gives run-time error because expected number of outputs are
7 \ not equal to actual number of outputs.
8
9 : DADD (S d1 d2 -- d3 ) D+ ;
10 \ Expected inputs and outputs are double numbers.
11
12 : EQUAL (S n1 n2 -- ? ) = ;
13 \ Expected output is a flag, but ?OUTPUTS only checks the
14 \ number of items on the stack, not their values.
15
```



uses a stack diagram with each definition.

The two words `?INPUTS` and `?OUTPUTS` are similar to `?ENOUGH` and could be used explicitly. But the point of this is to save beginners from making too many errors. By the time many beginners would become comfortable using these additional words, the need for them would have passed. That's why we need `IN/OUT` to make compilation of the error checker more automatic. They could also be used alone, much the same as `?ENOUGH` is used now, but that is still asking a lot from the beginner. In the accompanying code, you will notice that `;` is redefined to compile `?OUTPUTS` automatically. The input to `?OUTPUTS` is held on the

return stack at run time, to avoid interfering with data on the parameter stack. Note also that the right parenthesis ending a stack comment must be preceded by a blank, or there will be a "stack picture error."

The final step is to let the stack picture words themselves execute `IN/OUT`, compiling `?INPUTS` and `?OUTPUTS`. `(S` initializes the variables `IN-PUTS`, `OUTPUTS`, and `PARA-METERS` to zero, zero, and `INPUTS`, respectively. These variables keep tallies of the number of input and output parameters specified by the stack diagram. Then a loop is entered that reads the stack comment, adding the correct number of bytes to `INPUTS`, switching to `OUTPUTS`

when a `--` (double hyphen) is encountered, and tallying those until the `)` is seen. As here, only `n`, `d`, `?`, `-`, and `)` are recognized by the stack checker. Any other initial character causes a compile-time error.

Please don't get me wrong: I'm not suggesting that standard Forth programs should include run-time stack checking. I do think it is a good, temporary crutch for people who are frustrated by frequent and mysterious Forth system crashes. It also promotes good style, by encouraging the programmer to use stack diagrams to declare the inputs and outputs for each word they define. The cost in memory and speed are too great for anything but educational purposes.

Visit the **MACH 2 Product Support RoundTable™** on **GENie™** !!

# MACH 2

*Multi-tasking FORTH 83 Development System*

## **MACH 2 FOR THE MACINTOSH™**

**99.95**

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

## **MACH 2 FOR THE OS-9 OPERATING SYSTEM™**

**495.00**

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

## **MACH 2 FOR INDUSTRIAL BOARDS**

**495.00**

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

### **PALO ALTO SHIPPING COMPANY**

**P.O. Box 7430**

**Menlo Park, California 94026**

**415 / 854-7994 • 800 / 44FORTH**

VISA/MC accepted. CA residents include 6.5% sales tax.

Include shipping/handling with all orders: US \$6; Canada \$8; Europe \$25; Asia \$30

RoundTable and GENie are registered trademarks of the General Electric Information Services Company.

# PERPETUAL DATE ROUTINE

ALLEN ANWAY-SUPERIOR, WISCONSIN



A perpetual computing problem has been to come up with a perpetual calendar. The solution of W.C. Elmore from the *American Journal of Physics* is easy to program in fixed-point, double-precision Forth, the language automatically taking the required integer parts. But first, a little background to the calculation.

The kind of year you and I are most interested in is defined in terms of the seasons. Unfortunately, the lengths of the four seasons and the days are not evenly divisible. Julius Caesar defined the Julian calendar, with the astronomy of his time, to include a leap year every four years. By 1500 A.D., the calendar was ten days off from the seasons, so a calendar reform was pushed through by the prominent leader of the day, Pope Gregory XIII. He decreed that Thursday, October 4, 1582 would be followed by Friday, October 15, 1582, with a new set of rules for leap years:

year/4 evenly: leap year  
year/100 evenly: not leap year  
year/400 evenly: leap year

These rules are exact enough for the purpose. Protestant countries were less enthusiastic about the Pope's edict, England finally ruling that Wednesday, September 2, 1752 would be followed by Thursday, September 14, 1752. During the accompanying English riots, the people cried, "Give us back our 11 days!"

For a perpetual calendar, we define a unique day number for each day, consecutive calendar days having consecutive day numbers. Elmore has calculated the following:

Julian Day  $N = \text{Int}[365.25y] + \text{Int}[30.59(m-2)] + d + 30$

Gregorian Day  $N = \text{Int}[365.25y] + \text{Int}[y/400] - \text{Int}[y/100] + \text{Int}[30.59(m-2)] + d + 32$

assuming that January is month 13 of the previous year, and that February is month 14 of the previous year.

These two formulas give consecutive days across Pope Gregory's calendar gap. The 0.59 (month) fraction is any value between 7/12 and 6/10.

One calculates the weekday from:

$W = (N+3) \text{ mod } 7$   
with 0 = Sunday, 1 = Monday, 2 = Tuesday, ..., 6 = Saturday.

Accordingly, screen 77 presents the primitive Forth coding to find the Gregorian day and weekday. Note that the

program itself corrects for January and February month/year numbers. One can easily calculate the number of days between dates by taking the difference of Gregorian days. `UM/MOD I` have renamed `UM/MMOD` to show more consistently the beginning and ending results (`UD1 U2 -- U3 UD4`), where `U3` is the remainder and `UD4` is the quotient. Here is an example of program operation for New Year's Day 1985:

85 1 1 GREGDAY D.  
will print 725010

1985 1 1 GREGDAY D.  
will print 725010

725010.WEEKDAY .  
will print 2, for Tuesday.

I encourage the reader to take this primitive further, and to write an actual calendar.

*Allen Anway is a member of the American Association of Physics Teachers, and a computer coordinator for the University of Wisconsin at Superior. His eldest daughter is a third-generation physicist.*

```
SCR # 77
0 ( # 077 ) ( perpet calendar primitive )
1 FORTH-B3
2 ( year\month\day --- UD )
3 : GREGDAY
4 ROT ( m\d\y ) ( short year correct )
5 DUP 100 < IF 1900 + THEN
6 ROT ( d\y\m ) ( special, prev year )
7 DUP 3 < IF 12 + SWAP
8 1- SWAP THEN
9 2- 3059 100 U*/ ( month calculation )
10 ROT ( y\U\d ) ( day calculation )
11 32 + ( y\U ) 0 ( y\UD ) ROT ( UD\y )
12 DUP 100 / NEGATE SWAP ( UD\N\y )
13 DUP 400 / ROT + S->D ROT ( UD\N\y )
14 36525 UM* 100 UM/MMOD ( UD\N\mod\UD )
15 ROT DROP D+ D+ ; ( UD )
16
17 ( UD --- mod ) ( gregorian_day --- 0_6 )
18 : WEEKDAY 3 0 D+ 7 UM/MMOD 2DROP ;
19 ( 0=Sunday, 1=Monday, ... , 6=Saturday )
20 ;S
21
22 W.C.Elmore, Am. J. Phys. 44 482 (1976)
23 (May issue) adapted by Allen Anway
```

# CALL FOR PAPERS

*for the ninth annual*

## FORML CONFERENCE

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

**Following Thanksgiving, November 27-29, 1987**

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

### **Theme: Forth and the 32-bit Computer**

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

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

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

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

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

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

# HEADLESS COMPILER

DARREL JOHANSEN-REDWOOD CITY, CALIFORNIA

**L**ike many Forth metacompilers, this system allows the generation of headless Forth code. Headers are used by the Forth outer interpreter when searching the dictionary. Once a word is compiled into a new word in the Forth dictionary, only its CFA (code field) and its PFA (parameter field) are required for its execution. Its "head" — the LFA (link field address) and NFA (name field address) — are not used except for future compilations or for direct execution when encountered by **INTERPRET** in the terminal input buffer or when loading a Forth screen.

For words that may only be needed a few times in the definition of higher-level words, there is no reason to have their heads (link and name fields) clutter up the dictionary and take up valuable space. On the other hand, it may require more space to define a new set of words to "behead" them than would be gained by stripping them of name and link fields.

The ideal headless compiler could be loaded into any Forth system, could compile code with or without headers, then could be forgotten when it is no longer needed.

This metacompiler does all these functions. It uses standard Forth-79 syntax, and has been implemented for two different types of Forth dictionary structures: the Forth PADS system and the MVP-FORTH system for CP/M. With little or no modifications, it can be transported to any Forth-79 system.

The above two Forth systems differ in two ways that must be accommodated by the headless compiler. First, the LFAs of the two are in different places. In the PADS version, the LFA precedes the

NFA to speed up dictionary search times. Also in the PADS version, words below a certain point in the dictionary are searched before the rest of the dictionary. In the PADS version, virtual files are used in high memory and can cause serious system crashes if high memory is used in a scheme like this without paying extra-careful attention to the way virtual utilities and development aids use the same memory areas.

The headless compiler resides in high memory. It consists of three main sections:

1. The code for the new compiling words.
2. Space for storing the link and name field address that must be "pruned."
3. The headers for the headless words.

The use of the system is straightforward, and shouldn't cause any unusual side effects as long as a few things are remembered:

1. There are two areas of memory being used, both growing upward. Caution must be maintained with large systems so that the main dictionary does not creep up into the headless compiler. Also beware of using utilities in virtual files (such as **DUMP**) which may overwrite sections of the compiler in high memory. The editor cannot be used, nor can the assembler, programming aids, or utilities in the virtual files without a thorough understanding of the sections of memory which could conflict with the headless compiler.

2. The system must be pruned and unpatched before it can be saved. There cannot be any links to words in high memory beyond **HERE**.

3. The system cannot be simply

stripped back by **FORGET** before pruning without careful consideration.

4. The defining words **CREATE DOES>** do not function with the headless compiler and should be avoided or redefined to accommodate their special compiling functions.

5. A definition cannot be made that uses **[COMPILE]** to compile any of the new defining words (e.g., colon, tick, constant, variable, etc.). If these are required, they must be re-defined as a different word or patched into the words they are compiled in (like forward references in a metacompiler) at the end of compilation.

Most Forth metacompilers require that the entire dictionary be recompiled to get the advantage of a system with both heads and headerless words. If a system is to retain the basic Forth dictionary and its compiling words for the user, then very few, if any, of the low-level words will be removed from the dictionary. Only words that are to be taken out of the application for more efficient use of space and for security of the application code need to be stripped of their headers. This doesn't have to involve a new meta-compilation of the source code. In fact, the headless compiler can be invoked a number of times in the generation of the application. At each stage, a usable, extensible Forth system exists. The headless compiler might be used in a modular fashion, as subsystems are developed. These subsystems may be referenced later by a single word. The rest of the words in the subsystem probably won't be used by any other subsystem and can be stripped of their heads once they are compiled. A simple tool like a

fancy **VLIST** can list the CFAs of all headless words before they are pruned; if they are required in the future for some purpose, they can be executed and compiled even without their heads.

### Compilation Steps

1. Set initial **himem** so there is enough room for it to grow without crashing into the disk buffers or other Forth stuff kept at the very top of memory, but far enough above the main dictionary won't creep up into it.

2. Load the headerless compiler screens.

3. Load your application code, editing the words **HEADS-ON** and **HEADS-OFF** in the screen files to enter the compilation mode you desire. Normally, heads-on words are words that will be needed after this compilation process, either directly by the user in the application, or indirectly by words that will use them after the headless compiler has been pruned. The first and last word in this application should have their heads on unless you are very careful in pruning and unpatching. Also, don't recklessly **FORGET** the last word with heads above the headless words — this can mess up Forth. Usually, the word **MEND** is included as the last word of the metacompiler, just to avoid this situation.

4. When the application is loaded, type **HLIST** to get a printout of all the headerless words and their CFAs, in case you need them later.

5. Type **PRUNE** to unhook all the heads of the headerless words.

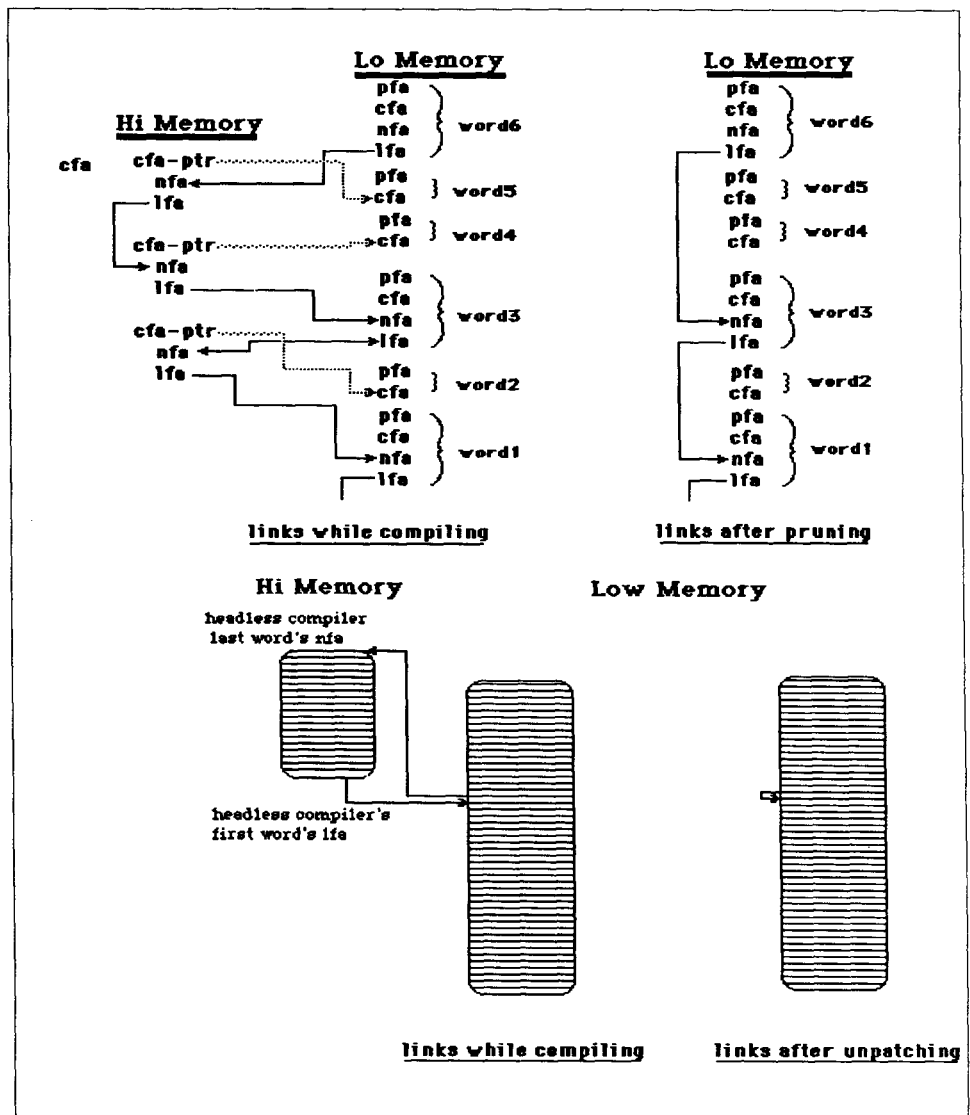
6. Type **UNPATCH** to unhook the new **INTERPRET** and re-establish the normal dictionary search order.

7. Enter **'META NFA 'MEND LFA !** to cut away the headless compiler. **META** is the last word defined before the headless compiler was loaded. **MEND** is the last word in the metacompiler and has a regular head.

8. Type **FREEZE COLD** and then save the system.

### Limitations

**DOES>** will not work. A new version of **DOES>** must be generated in



```

SCR #1
0 ( HEADLESS COMPILER -1 **IBM SYSTEM** )
1 FORGET TARGET ( strip back our system )
2
3 FORGET-SYS ( get all virtual systems out )
4
5 HERE
6 DOOO DP ! VARIABLE DPBODY DPBODY ! ( from here above )
7
8 VARIABLE H/B 1 H/B ! ( 0 = w/heads 1 = w/o heads )
9
A VARIABLE DPHEAD 0 CONSTANT HEADLESS? VARIABLE TOP
B
C
D
-->

SCR #2
0 ( HEADLESS COMPILER -2 **IBM SYSTEM** )
1 VARIABLE FIELDS 6 ALLGOT
2
3 : HLIST [COMPILE] ? DUP 14F5 = IF DROP CONTEXT @ @ ELSE NFA THEN
4 CR " PFA-PTR CFADR NAME" CR BEGIN 0
5 FIELDS ! DUP PFA 2- DUP FIELDS 2+ ! DUP TOP @ SWAP UK FIELDS !
6 @ FIELDS 4 + ! FIELDS @ IF DUP FIELDS 2+ @ @ 0 6 D,R FIELDS 4 +
7 @ 0 8 D,R 4 SPACES ID, CR THEN PFA-LFA @ DUP [ DPBODY @ ]
8 LITERAL UK PAUSE ?TERMINAL OR UNTIL DROP ;
9
A
B HLIST will print out the list of words to be beheaded with their
C current PFA-PTR and their CFA's in the main dictionary. This
D address can be used later for compiling or executing the
E headless word. **** EXECUTE or **** ?
F HLIST is adapted from decompiling VLIST for PADS

```

the metacompiler, if it is required.

**FORGET** will not forget gracefully through the headless words. Use with caution.

' (tick) will work in the compiling and execution modes, but it cannot be compiled in definitions like:

```
:TEST... [COMPILE]' ...;
```

This is generally true of all new defining words, since the CFA of the new word would be compiled into the definition. This would crash the system after the heads are wiped out.

**CODE** is not implemented for headless words.

#### CP/M on IBM with Baby Blue

Since the memory space available for CP/M (Baby Blue CP/M board on the IBM PC) is greater than for normal CP/M systems, this extra memory can be used by the metacompiler. The metacompiler can be compiled in a lower part of the Forth system, then saved as a virtual file, reloaded when needed, and

stripped away after headless compilation. The following steps are required:

1. Set the new **DPHEAD** to **LIMIT**, and disable the stack check word **?STACK** with

```
' EXIT CFA ' ?STACK !
```

Then compile the metacompiler as normal.

2. Save the metacompiler as a virtual file by figuring out how many 1K blocks of memory it uses (usually between one and two) and save it to some unused Forth screens with:

```
HEX LIMIT N BLOCK 400
```

```
CMOVE
```

```
LIMIT 400 + N 1+
```

```
BLOCK400 CMOVE
```

(where N is the first screen)

3. Then do the normal **PRUNE**, **UNPATCH**, and **' META NFA '** **MEND LFA !** to unpatch it.

4. Restore **?STACK** with **' SP@ CFA ' ?STACK !**

5. Now the virtual file can be moved into memory, used, and then unpatched

## ADVERTISERS INDEX

Bryte - 16

Click Software - 18, 19

Computer Cowboys - 9

Dash, Find & Associates - 10

FORML - 35

FORTH, Inc. - 8, 41

Forth Interest Group - 21-24, 44

Hampton Corp. - 10

Harvard Softworks - 7

Inner Access - 38

Laboratory Microsystems - 11

MCA - 25

MicroMotion - 26

Miller Microcomputer Services - 28

Mountain View Press - 39

Next Generation Systems - 6

Palo Alto Shipping Co. - 33

Prentice-Hall - 13

Software Composers - 2

## HANDS-ON FORTH WORKSHOPS

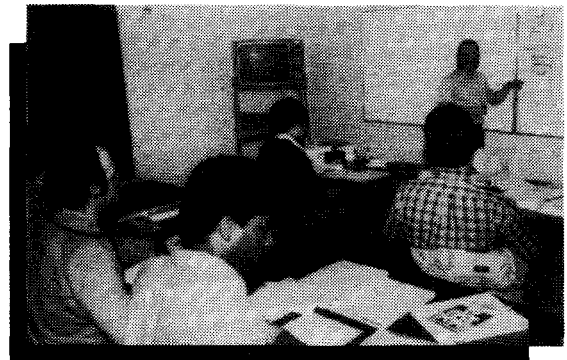
Get hands-on experience using FORTH for control applications with your own Super-8 20Mhz controller that you can take back with you. Benefit from the experienced teachers who have taught FORTH since 1980 to employees of IBM, HP, EG&G, ATARI, MARATHON, SANDIA LABS and many other companies. This course is suitable for beginning and intermediate FORTH programmers alike. No prerequisites necessary.

DATES: JUNE 15-19, 1987 9:00-4:00

JULY 27-31, 1987 9:00-4:00

PRICE: \$850 Includes all materials, textbook and Super-8 FORTH controller.

**Call or write for details on the workshops at (415) 574-8295**



FORTH WORKSHOP instructors Gary Feierbach and Paul Thomas, co-authors of "FORTH TOOLS and APPLICATIONS" have taught FORTH workshops since 1980. Gary and Paul have written numerous FORTH compilers and interpreters and many large FORTH applications.



# Inner Access

1155-A Chess Drive, Suite D, Foster City, CA 94404 · (415) 574-8295

## FORTH SOURCE™

### WISC CPU/16

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

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

### MVP-FORTH

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

### MVP Books - A Series

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

### MVP-FORTH Software - A trans-portable FORTH

<input type="checkbox"/> MVP-FORTH Programmer's Kit including disk, documentation. Volumes 1, 2 & 7 of MVP Series, FORTH Applications, and Starting FORTH, IBM, Apple, Amiga, CP/M, MS-DOS, PDP-11 and others. Specify.	\$175
<input type="checkbox"/> MVP-FORTH Enhancement Package for IBM Programmer's Kit. Includes full screen editor & MS-DOS file interface.	\$110
<input type="checkbox"/> MVP-FORTH Floating Point and Math <input type="checkbox"/> IBM, <input type="checkbox"/> Apple, or <input type="checkbox"/> CP/M, 8".	\$75
<input type="checkbox"/> MVP-LIBFORTH for IBM. Four disks of enhancements.	\$25
<input type="checkbox"/> MVP-FORTH Screen editor for IBM.	\$15
<input type="checkbox"/> MVP-FORTH Graphics Extension for <input type="checkbox"/> IBM or <input type="checkbox"/> Apple	\$80
<input type="checkbox"/> MVP-FORTH PADS (Professional Application Development System) An integrated system for customizing your FORTH programs and applications. PADS is a true professional development system. Specify Computer: <input type="checkbox"/> IBM <input type="checkbox"/> Apple	\$500
<input type="checkbox"/> MVP-FORTH MS-DOS file interface	\$80
<input type="checkbox"/> MVP-FORTH Floating Point Math	\$100
<input type="checkbox"/> MVP-FORTH Graphics Extension	\$80
<input type="checkbox"/> MVP-FORTH EXPERT-2 System for learning and developing knowledge based programs. Specify <input type="checkbox"/> Apple, <input type="checkbox"/> IBM, or <input type="checkbox"/> CP/M 8".	\$100

Order Numbers:  
**800-321-4103**

(In California) 415-961-4103

**FREE  
CATALOG**

**MOUNTAIN VIEW  
PRESS**

PO BOX X  
Mountain View, CA 94040

```

SCR #3
0 ( HEADLESS COMPILER -3 IBM)
1 VARIABLE TWIG-PTR 0 TWIG-PTR ! ( ptr at current twig)
2 CREATE TWIGS 200 ALLOT ( table of twigs to be trimmed)
3
4 : +TWIG TWIG-PTR @ DUP 1FA > ( --adr) ( get current twig addr)
5 IF ." twig ref.blk ov" ABORT THEN TWIGS + UPDATE ;
6
7 : CLR-TWIG TWIGS 100 0 FILL ; CLR-TWIG ( clear table )
8 : TWIG-TBL! +TWIG ! 2 TWIG-PTR +! ; ( store current twig)
9 : DPHEAD HERE DPBODY ! DPHEAD @ DP ! ;
A : SETBODY HERE DPHEAD ! DPBODY @ DP ! ;
B : HEADS-ON 0 ? HEADLESS? ! H/B @ 0= IF 1 H/B !
C : HERE TWIG-TBL! THEN ;
D : HEADS-OFF 1 ? HEADLESS? ! H/B @ IF 0 H/B !
E : LATEST TWIG-TBL! THEN ;
F --->

SCR #4
0 ( HEADLESS COMPILER -4 IBM)
1
2 : UNPATCH < INTERPRET > 2- ( cfa) ? INTERPRET !
3 ' FORTH ? -FIND 10 + ! CLR-TWIG ( FKEY) 0 TWIG-PTR ! ;
4 ( restore to normal INTERPRET, set PADS dict search order)
5
6 : PRUNE 0 TWIG-PTR ! ( Cut away heads)
7 BEGIN +TWIG @ ( get nfa of last regular head)
8 +TWIG 2+ @ ( get LFA of next regular head)
9 DUP 0= IF EXIT THEN ( end of buffer?)
A DUP @ TOP @ UK ABORT" Twig res error" ( err chk)
B ! ( store into LFA of next regular head)
C 4 TWIG-PTR +! ( move pointer up to next field)
D AGAIN ; ( continue until finding a 0)
E
F --->

SCR #5
0 ( HEADLESS COMPILER -5 IBM )
1
2 : CREATE-HEAD >IN @ SETHEAD FIRST 190 - HERE UK
3 IF CR 7 EMIT ." close to buff" CR ( space available?)
4 THEN LATEST , BL WORD DUP 1+ C@ 0 =
5 IF -2 DP +! 1 ELSE 0
6 THEN ABORT" null?" DUP CONTEXT @ @ 1 <<FIND>
7 IF DDROP WARNING @
8 IF CR DUP COUNT TYPE SPACE ." headless,not unique " THEN
9 THEN DUP C@
A WIDTH MIN 1+ ALLOT DUP 80 TOGGLE HERE 1-
B 80 TOGGLE DPBODY @ , CURRENT @ ! SETBODY
C >IN @ SWAP -- +>IN ! ; ( move input stream around head)
D
E
F --->

SCR #6
0 ( HEADLESS COMPILER -6 IBM )
1 ' 1 2- ( cfa) @ CONSTANT *CONSTANT* ( Get cfa's of )
2 ' DPHEAD 2- @ CONSTANT *VARIABLE* ( regular words)
3 ' QUIT 2- @ CONSTANT *COLON*
4 ' TWIGS 2- @ CONSTANT *CREATE*
5
6 : CONSTANT HEADLESS? IF >IN @ CREATE-HEAD
7 *CONSTANT* . ELSE CONSTANT THEN ;
8 : VARIABLE HEADLESS? IF >IN @ CREATE-HEAD ;
9 *VARIABLE* , 0 ELSE VARIABLE THEN ;
A : : SP@ CSP ! HEADLESS? IF >IN @ CREATE-HEAD
B *COLON* , SMUDGE 1 ELSE ; THEN ;
C : CREATE HEADLESS? IF >IN @ CREATE-HEAD
D *CREATE* , ELSE CREATE THEN ;
E ' HERE NFA ' -FIND 10 + ! ( pads patch for dict search)
F --->

SCR #7
0 ( HEADLESS COMPILER -7 IBM )
1 : TIC [COMPILE] ? ; IMMEDIATE ( old tick)
2 : ' -FIND 0= ABORT" nt fnd" DROP TOP @ OVER UK IF
3 2- @ 2+ THEN [COMPILE] LITERAL ; IMMEDIATE
4 : <INTERPRET BEGIN -FIND ( in dictionary?)
5 IF STATE @ < ( compiling?)
6 IF 2- ( cfa) TOP @ OVER UK IF @ THEN
7 ELSE 2- ( cfa) TOP @ OVER UK IF @ THEN EXECUTE
8 THEN
9 ELSE HERE NUMBER DPL @ 1+ ( then try to put in as number)
A IF [COMPILE] DLITERAL
B ELSE DROP [COMPILE] LITERAL THEN
C THEN ?STACK AGAIN ; HERE TOP !
D TIC <INTERPRET 2- ( cfa) TIC INTERPRET ! ( IBM version)
E ( no new words can be compiled into meta system from here)
F HEADS-ON SETBODY CREATE MEND ( optional marker)

SCR #8
0 ( HEADLESS COMPILER -1 **CFM SYSTEM** )
1 FORGET TARGET ( strip back our system)
2 ' EXIT 2- ' ?STACK ! ( include for batch)
3
4 CREATE META 4 ALLOT ( optional marker)
5
6 ( HERE LIMIT DP ! ) ( for batch w/ baby blue card in IBM)
7 HERE AOOD DP ! ( for regular cp/m system)
8
9 VARIABLE DPBODY DPBODY ! ' DPBODY LFA META !
A VARIABLE H/B 1 H/B ! ( 0 = w/heads 1 = w/o heads)
B VARIABLE DPHEAD VARIABLE TOP
C
D 0 CONSTANT HEADLESS?
E
F --->

```

without recompiling every time.

6. One additional feature is added in this version. The NFA of the last word in the metacompiler must be saved so that the LFA of **MEND** can be patched in when it is reloaded. The word **META** allots a couple of extra bytes to save this address.

Here is a list of newly defined words and their functions in the headless compiler:

**HEADS-ON** (--)

Starts the normal compilation of words into the Forth dictionary.

**HEADS-OFF** (--)

Starts the compilation of headerless words into the Forth dictionary.

**CREATE-HEAD** (--)

Creates a head in high memory for a word which will be headless in the main dictionary.

**VARIABLE** (--)

Creates a variable without a head if **HEADS-OFF** has been previously set. Otherwise, creates a normal variable.

**CONSTANT** (--)

Creates a constant without a head if **HEADS-OFF** has been previously set. Otherwise, creates a normal constant.

**:** (--)

Creates a colon-defined word without a head if **HEADS-OFF** has been previously set. Otherwise, creates a normal colon-defined word.

**CREATE** (--)

Creates a word without a head if **HEADS-OFF** has been previously set. Otherwise, creates a normal word.

**'** (-- pfa)

Gets the PFA of the following word in the input stream. Does **CFA @ 2+** if the word is in himem.

**DPHEAD** (--)

Variable containing current position of DP in himem.

**DPBODY** (--)

```
SCR #?
0 ( HEADLESS COMPILER -2 CPM SYSTEM )
1
2 VARIABLE FIELDS 6 ALLOT ( storage for hlist )
3
4 : HLIST CR ." CFA-PTR PFA NAME" CR 0 FIELDS ! CONTEXT @ @
5 BEGIN PFA DUP DUP FIELDS 4 + !
6 2- @ FIELDS 2 + ! TOP @ OVER UK FIELDS !
7 FIELDS @ IF FIELDS 2+ @ 0 & D.R FIELDS 4 + @ 0 @ D.R 4
8 SPACES DUP NFA ID. CR THEN
9 4 - @ DUP [ DPBODY @ ] LITERAL UK
A PAUSE ?TERMINAL OR UNTIL DROP ;
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

SCR #A
0 ( HEADLESS COMPILER -3 CPM )
1
2 VARIABLE TWIG-PTR ( ptr at current twig )
3
4 CREATE TWIGS 100 ALLOT ( table of twigs to be trimmed )
5
6 +TWIG TWIG-PTR @ DUP FA > ( --adr ) ( get current twig addr )
7 IF ." twig ref.blk ov" ABORT THEN TWIGS + UPDATE ;
8
9 : CLR-TWIG 0 TWIG-PTR ! TWIGS 100 0 FILL ; ( clear table )
A CLR-TWIG
B
C : TWIG-TBL ! +TWIG ! 2 TWIG-PTR + ! ; ( store current twig )
D
E : UNPATCH ' <INTERPRET> 2- 'INTERPRET ! CLR-TWIG ;
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

SCR #B
0 ( HEADLESS COMPILER -4 CPM )
1 : SETHEAD HERE DPBODY ! DPHEAD @ DP ! ;
2
3 : SETBODY HERE DPHEAD ! DPBODY @ DP ! ;
4
5 : HEADS-ON 0 ' HEADLESS? ! H/B @ 0= IF 1 H/B !
6 HERE TWIG-TBL ! THEN ;
7
8 : HEADS-OFF 1 ' HEADLESS? ! H/B @ IF 0 H/B !
9 LATEST TWIG-TBL ! THEN ;
A
B : CREATE-HEAD >IN @ SETHEAD BL WORD
C DUP C@ WIDTH @ MIN 1+ ALLOT
D DUP 80 TOGGLE HERE 1-- 80 TOGGLE
E LATEST ( cpm lfa here ) DPBODY @ , CURRENT @ ! SETBODY
F >IN @ SWAP - + >IN ! ;
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

SCR #C
0 ( HEADLESS COMPILER -5 CPM PRUNE )
1
2 : PRUNE 0 TWIG-PTR !
3 BEGIN +TWIG @ ( get nfa of last regular head )
4 +TWIG 2+ @ ( get LFA of next regular head )
5 DUP 0= IF EXIT THEN ( end of buffer? )
6 DUP @ IF AND + 1+ ( traverse name field to lfa )
7 DUP @ TOP @ UK ABORT" twig res error"
8 4 TWIG-PTR + ! ( store into LFA of next regular head )
9 AGAIN ; ( move pointer up to next field )
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

SCR #D
0 ( HEADLESS COMPILER -6 CPM )
1
2 : DPHEAD 2-- ( cfa ) @ CONSTANT *CONSTANT* ( Get cfa's of )
3 : GUIT 2-- @ CONSTANT *VARIABLE* ( regular words )
4 : TWIGS 2-- @ CONSTANT *COLON*
5 : TWIGS 2-- @ CONSTANT *CREATE*
6
7 : CONSTANT HEADLESS? IF >IN @ CREATE-HEAD
8 *CONSTANT* ELSE CONSTANT THEN ;
9 : VARIABLE HEADLESS? IF >IN @ CREATE-HEAD
A *VARIABLE* 0 , ELSE VARIABLE THEN ;
B : : SP@ CSP ! HEADLESS? IF >IN @ CREATE-HEAD
C *COLON* , SMUDGE ] ELSE ; THEN ;
D : CREATE HEADLESS? IF >IN @ CREATE-HEAD
E *CREATE* , ELSE CREATE THEN ;
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

SCR #E
0 ( HEADLESS COMPILER -7 CPM )
1 : TIC [COMPILE] ' ; IMMEDIATE ( old tick )
2 : ' -FIND 0= ABORT" nt fnd" DROP TOP @ OVER UK IF
3 2- @ 2+ THEN [COMPILE] LITERAL ; IMMEDIATE
4 : <INTERPRET BEGIN -FIND ( in dictionary? )
5 IF STATE @ < ( compiling? )
6 IF 2- TOP @ OVER UK IF @ THEN ( @ if headless )
7 ELSE 2- TOP @ OVER UK IF @ THEN EXECUTE ( ditto )
8 THEN
9 ELSE HERE NUMBER DPL @ 1+ ( then try to put in as number )
A IF [COMPILE] DLITERAL
B ELSE DROP [COMPILE] LITERAL THEN ( no stack chk for )
C THEN ( ?STACK ) AGAIN ; HERE TOP ! ( baby blue on ibm )
D TIC <INTERPRET DUP 2- 'INTERPRET ! NFA META 2+ !
E ( no new words can be compiled into meta system from here )
F HEADS-ON SETBODY CREATE MEND
```



## Work at the "cutting edge"!

Join the programming team at FORTH, Inc. using our powerful multi-user polyFORTH software to develop challenging real-time applications such as:

- A 400-computer network controlling an entire airport
- A multiprocessor control system for a major automobile manufacturer
- The cell controller for a semiconductor process cell
- Expert systems for real-time diagnostics

We have *permanent, full-time programming positions* open at several levels, at our Manhattan Beach headquarters. Send us your resume today if you have:

- Engineering training or experience, and
- Written at least one real-time application in Forth

Plus at least one of the following:

- Assembler-level experience with two or more processors
- Managed several Forth programmers
- Good writing and/or teaching skills

If your present job doesn't provide enough challenge, variety and professional growth, you may find the stimulating environment at FORTH, Inc. ideal for you.

Send your resume today to: FORTH, Inc., 111 N. Sepulveda Blvd., Manhattan Beach, CA 90266. FORTH, Inc. is an equal opportunity employer.



Variable containing current position of DP in main dictionary.

**TIC** (-- pfa)

Gets PFA of a word with a normal head. This is the old ' and is used mainly for patching <INTERPRET.

<INTERPRET

Interprets or compiles words, depending on **STATE**. If the word is found to have a CFA at an address above a certain point, it is a headerless word and its CFA is gotten with **CFA @**.

**TWIGS** (-- adr)

An array in memory that records each time **HEADS-ON** or **HEADS-OFF** is switched. **DP** and **LATEST** at that point in time are stored in the array, and later used to behead the headless words.

**TWIG-PTR** (-- adr)

A variable containing the next position in **TWIGS** to be used.

**HLIST**

A routine to list all headless words in the dictionary (before pruning, of course) along with their CFAs. The CFAs can be used later for compilation or execution.

**PRUNE**

A routine that strips away all the heads in high memory and resets the links in the main dictionary, so headerless words are not seen by **FIND**.

**UNPATCH**

A routine to reset the normal **INTERPRET** and, in the PADS system, to re-establish the previous dictionary search order.

**References**

*MVP-FORTH Professional Applications Development System (PADS)*, Tom Wempe. FORTHKIT, 240 Prince Street, Los Gatos, CA 95030.

*META-FORTH™ A Metacompiler for FIG-FORTH*, John J. Cassidy. 11 Mira Monte Road, Orinda, CA 94563.

*Systems Guide to fig-FORTH*, C.H. Ting. Zero Edition, Offete Enterprises, Inc., 1306 South B Street, San Mateo, CA 94402.

*All About Forth*, 2nd ed., Glen B.

Haydon. Mountain View Press, Inc., P.O. Box 4656, Mountain View, CA 94040.

*RSC-FORTH User's Manual*, Rockwell International Corporation. 1983, Document No. 29651N51.

*MetaFORTH for the 6502*, Darrel Johansen. 1983, San Francisco, CA.

Thanks to Serge Modular Music Systems and to Orion Instruments for time, encouragement, and an application to use this project on.

*Darrel Johansen has been programming in Forth for ten years, including projects in electronic music synthesizer control. He is currently senior engineer at Orion Instruments, Inc.*

# FIG CHAPTERS

## U.S.A.

### • ALABAMA

**Huntsville FIG Chapter**  
Tom Konantz (205) 881-6483

### • ALASKA

**Kodiak Area Chapter**  
Horace Simmons (907) 486-5049

### • ARIZONA

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

### • ARKANSAS

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

### • CALIFORNIA

**Los Angeles Chapter**  
4th Sat., 10 a.m.  
Hawthorne Public Library  
12700 S. Grevillea Ave.  
Phillip Wasson (213) 649-1428  
**Monterey/Salinas Chapter**  
Bud Devins (408) 633-3253  
**Orange County Chapter**  
4th Wed., 7 p.m.  
Fullerton Savings  
Huntington Beach  
Noshir Jesung (714) 842-3032  
**San Diego Chapter**  
Thursdays, 12 noon  
Guy Kelly (619) 450-0553  
**Sacramento Chapter**  
4th Wed., 7 p.m.  
1798-59th St., Room A  
Tom Ghormley (916) 444-7775  
**Silicon Valley Chapter**  
4th Sat., 10 a.m.  
H-P, Cupertino  
George Shaw (415) 276-5953  
**Stockton Chapter**  
Doug Dillon (209) 931-2448

### • COLORADO

**Denver Chapter**  
1st Mon., 7 p.m.  
Steven Sams (303) 477-5955

### • CONNECTICUT

**Central Connecticut Chapter**  
Charles Krajewski (203) 344-9996

### • FLORIDA

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

### • GEORGIA

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

### • ILLINOIS

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

### • INDIANA

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

### • IOWA

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

### Central Iowa FIG Chapter

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

### • KANSAS

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

### • MASSACHUSETTS

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

### • MICHIGAN

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

### • MINNESOTA

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

### • MISSOURI

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

### • NEW JERSEY

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

### • NEW MEXICO

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

### • NEW YORK

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

### • NORTH CAROLINA

**Raleigh Chapter**  
Frank Bridges (919) 552-1357

### • OHIO

**Akron Chapter**  
3rd Mon., 7 p.m.  
McDowell Library  
Thomas Franks (216) 336-3167  
**Athens Chapter**  
Isreal Urieli (614) 594-3731  
**Cleveland Chapter**  
4th Tues., 7 p.m.  
Chagrin Falls Library  
Gary Bergstrom (216) 247-2492  
**Dayton Chapter**  
2nd Tues. & 4th Wed., 6:30 p.m.  
CFC, 11 W. Monument Ave.,  
#612  
Gary Granger (513) 257-6984

### • OKLAHOMA

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

### • OREGON

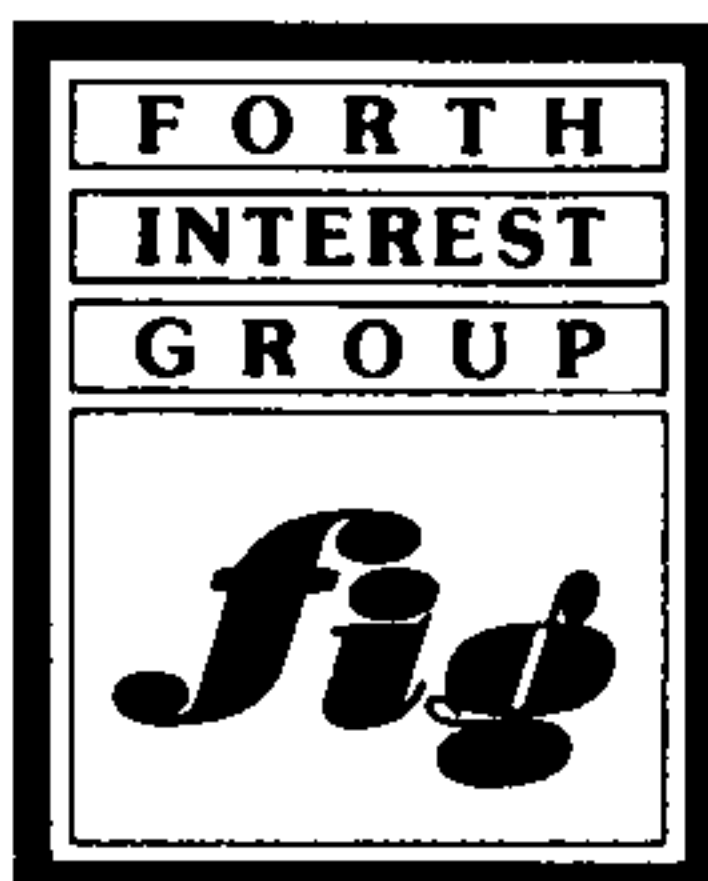
**Greater Oregon Chapter**  
Beaverton

- 2nd Sat., 1 p.m.  
Tektronix Industrial Park,  
Bldg. 50  
Tom Almy (503) 692-2811  
**Willamette Valley Chapter**  
4th Tues., 7 p.m.  
Linn-Benton Comm. College  
Pann McCuaig (503) 752-5113
- **PENNSYLVANIA**  
**Philadelphia Chapter**  
4th Sat., 10 a.m.  
Drexel University, Stratton Hall  
Melanie Hoag (215) 895-2628
  - **TENNESSEE**  
**East Tennessee Chapter**  
Oak Ridge  
2nd Tues., 7:30 p.m.  
Sci. Appl. Int'l. Corp., 8th Fl.  
800 Oak Ridge Turnpike,  
Richard Secrist (615) 483-7242
  - **TEXAS**  
**Austin Chapter**  
Contact Matt Lawrence  
P.O. Box 180409  
Austin, TX 78718  
**Dallas/Ft. Worth**  
**Metroplex Chapter**  
4th Thurs., 7 p.m.  
Chuck Durrett (214) 245-1064  
**Houston Chapter**  
1st Mon., 7 p.m.  
Univ. of St. Thomas  
Russel Harris (713) 461-1618  
**Perliman Basin Chapter**  
Odessa  
Carl Bryson (915) 337-8994
  - **UTAH**  
**North Orem FIG Chapter**  
Contact Ron Tanner  
748 N. 1340 W.  
Orem, UT 84057
  - **VERMONT**  
**Vermont Chapter**  
Vergennes  
3rd Mon., 7:30 p.m.  
Vergennes Union High School  
Rm. 210, Monkton Rd.  
Don VanSyckel (802) 388-6698
  - **VIRGINIA**  
**First Forth of Hampton**  
**Roads**  
William Edmonds (804) 898-4099  
**Potomac Chapter**  
Arlington  
2nd Tues., 7 p.m.  
Lee Center  
Lee Highway at Lexington St.  
Joel Shprentz (703) 860-9260  
**Richmond Forth Group**  
2nd Wed., 7 p.m.  
154 Business School  
Univ. of Richmond  
Donald A. Full (804) 739-3623
  - **WISCONSIN**  
**Lake Superior FIG Chapter**  
2nd Fri., 7:30 p.m.  
Main 195, UW-Superior  
Allen Anway (715) 394-8360  
**MAD Apple Chapter**  
Contact Bill Horton  
502 Atlas Ave.  
Madison, WI 53714  
**Milwaukee Area Chapter**  
Donald Kimes (414) 377-0708
- INTERNATIONAL**
- **AUSTRALIA**  
**Melbourne Chapter**  
1st Fri., 8 p.m.  
Contact Lance Collins  
65 Martin Road  
Glen Iris, Victoria 3146  
03/29-2600  
**Sydney Chapter**  
2nd Fri., 7 p.m.  
John Goodsell Bldg., Rm. LG19  
Univ. of New South Wales  
Contact Peter Tregeagle  
10 Binda Rd., Yowie Bay  
02/524-7490
  - **BELGIUM**  
**Belgium Chapter**  
4th Wed., 20:00h  
Contact Luk Van Look  
Lariksdreff 20  
2120 Schoten  
03/658-6343  
**Southern Belgium Chapter**  
Contact Jean-Marc Bertinchamps  
Rue N. Monnom, 2  
B-6290 Nalinnes  
071/213858
  - **CANADA**  
**Northern Alberta Chapter**  
4th Sat., 1 p.m.  
N. Alta. Inst. of Tech.  
Tony Van Muyden (403) 962-2203  
**Nova Scotia Chapter**  
Halifax  
Howard Harowitz (902) 477-3665  
**Southern Ontario Chapter**  
Quarterly, 1st Sat., 2 p.m.  
Genl. Sci. Bldg., Rm. 212  
McMaster University  
Dr. N. Soltseff (416) 525-9140  
ext. 3  
**Toronto Chapter**  
Contact John Clark Smith  
P.O. Box 230, Station H  
Toronto, ON M4C 5J2  
**Vancouver Chapter**  
Don Vanderweele (604) 941-4073
  - **COLOMBIA**  
**Colombia Chapter**  
Contact Luis Javier Parra B.  
Apto. Aereo 100394  
Bogota 214-0345
  - **DENMARK**  
**Forth Interesse Gruupe**  
**Denmark**  
Copenhagen  
Erik Oestergaard, 1-520494
  - **ENGLAND**  
**Forth Interest Group- U.K.**  
London  
1st Thurs., 7 p.m.  
Polytechnic of South Bank  
Rm. 408  
Borough Rd.  
Contact D.J. Neale  
58 Woodland Way  
Morden, Surry SM4 4DS
  - **FRANCE**  
**French Language Chapter**  
Contact Jean-Daniel Dodin  
77 Rue du Cagire  
31100 Toulouse  
(16-61)44.03.06  
**FIG des Alpes Chapter**  
Annely  
Georges Seibel, 50 57 0280
  - **GERMANY**  
**Hamburg FIG Chapter**  
4th Sat., 1500h  
Contact Horst-Gunter Lynsche  
Common Interface Alpha  
Schanzenstrasse 27  
2000 Hamburg 6
  - **HOLLAND**  
**Holland Chapter**  
Contact Adriaan van Roosmalen  
Heusden Houtsestraat 134  
4817 We Breda  
31 76 713104
  - **IRELAND**  
**Irish Chapter**  
Contact Hugh Dobbs  
Newton School  
Waterford  
051/75757 or 051/74124
  - **ITALY**  
**FIG Italia**  
Contact Marco Tausel  
Via Gerolamo Forni 48  
20161 Milano  
02/435249
  - **JAPAN**  
**Japan Chapter**  
Contact Toshi Inoue  
Dept. of Mineral Dev. Eng.  
University of Tokyo  
7-3-1 Hongo, Bunkyo 113  
812-2111 ext. 7073
  - **NORWAY**  
**Bergen Chapter**  
Kjell Birger Faeraas, 47-518-7784
  - **REPUBLIC OF CHINA**  
**(R.O.C.)**  
Contact Ching-Tang Tzeng  
P.O. Box 28  
Lung-Tan, Taiwan 325
  - **SWEDEN**  
**Swedish Chapter**  
Hans Lindstrom, 46-31-166794
  - **SWITZERLAND**  
**Swiss Chapter**  
Contact Max Hugelshofer  
ERNI & Co., Elektro-Industrie  
Stationsstrasse  
8306 Bruttisellen  
01/833-3333
- SPECIAL GROUPS**
- **Apple Corps Forth Users Chapter**  
1st & 3rd Tues., 7:30 p.m.  
1515 Sloat Boulevard, #2  
San Francisco, CA  
Dudley Ackerman  
(415) 626-6295
  - **Baton Rouge Atari Chapter**  
Chris Zielewski (504) 292-1910
  - **FIGGRAPH**  
Howard Pearlmutter  
(408) 425-8700
  - **NC4000 Users Group**  
John Carpenter (415) 960-1256

**NOW AVAILABLE**

**1986 FORML  
CONFERENCE  
PROCEEDINGS**

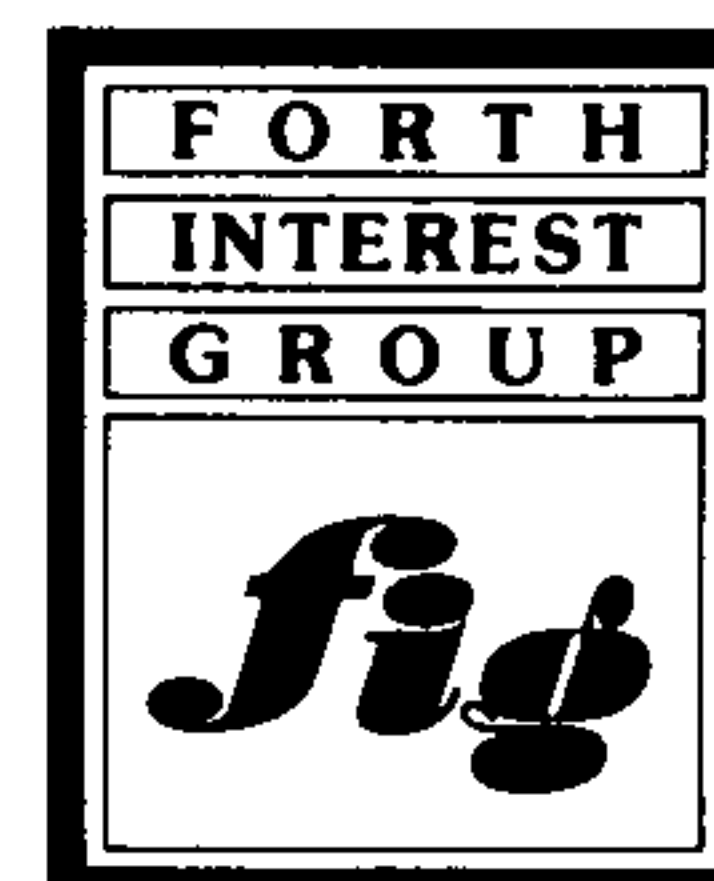
Eighth Asilomar FORML Conference  
November 28-30, 1986  
Asilomar Conference Center,  
Pacific Grove, California



---

**\$30 EACH**

---



**FROM THE FORTH INTEREST GROUP**

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

Bulk Rate  
U.S. Postage  
PAID  
Permit No. 3107  
San Jose, CA

Address Correction Requested