Serial I/O & Interrupts

Fast Fixed-Point Trig

Case Conversion in KEY

Select, Ordered, Perform

Moore Chats on CompuServe
Mach1™

Multi-tasking FORTH-83 Development Systems for 68000 and 68020-based microcomputers and industrial target boards.

Mach1 is a FAST, 32-bit subroutine-threaded implementation of FORTH. All Mach1 systems include:

- Unlimited multi-tasking--any number of background/terminal tasks are allowed.
- Local variables for readable, recursive, re-entrant programming.
- Standard text files--Any text-only editor/word processor may be used.
- A STANDARD Motorola 68000 assembler (infix) which supports forward label references.

### Mach1 on the Mac features:

- EASY creation of stand-alone applications
- Complete toolbox support (including Mac Plus routines)
- Macintalk speech driver support
- Redirection of I/O to serial ports/devices
- Graphics printing support
- 80-bit SANE floating point
- 68020 compatible

*comes with Switcher/Edit and 400 page manual*

### Mach1 on the ST features:

- EASY creation of stand-alone applications
- Full GEM and DOS support
- Integrated GEM editor
- 68020 compatible

*comes with 300 page manual*

### Mach1 in EPROM:

- 16K FORTH Kernel
- 16K 68000 Assembler
- 16K Disassembler/Debugger

*Call for source licensing arrangements...*

*Executes sieve on 16MHz 68020 in 1.3 seconds*

---

Apple Macintosh™

$49.95

$10 Demo Disk also available

Atari ST™

$59.95

$10 Demo Disk also available

Commodore Amiga™

($59.95 Available September 1986)

Industrial Target Systems

Palo Alto Shipping Company • P.O. Box 7430 • Menlo Park, California 94026

(800) 44FORTH - Sales • (415) 854-7994 - Support

VISA/MC Accepted • Include S/H on all orders ($5 US/Canada, $10 overseas) • CA res. add 6.5% tax
FEATURES

8 Interrupt-Driven Serial Input
by John S. James
The difficulty of implementing interrupt-driven serial I/O has caused a bottleneck in programming for PC compatibles. This article provides a working example of serial input, and a good deal of advice to get you over the hurdles.

14 Fast Fixed-Point Trig
by Johann Borenstein
The author’s sine routine uses optimal scaling of series parameters to eliminate all divisions. It runs nine times faster than Bumgarner’s version, at a mild sacrifice in precision: a useful routine for many applications.

21 Case Conversion in KEY
by David W. Harralson
Most keyboards return the lower-case value of a pressed key unless shift-lock is pressed, and on older terminals you cannot then enter numbers without taking off the shift-lock. fig-Forth can perform case conversion automatically, with the code this author presents.

22 Select, Ordered, Perform
by Wil Baden
One application of the Quicksort technique is based on the “Simple Files” example in Starting Forth. It allows easy selection and display of sorted records. Three end-user words are provided to accomplish the basic record manipulations.

23 TI 99/4A ISR Installation
by Gene Thomas
Interrupt service routines are fast routines which are performed in 1/60 second or less. While operating in Forth, user-defined ISRs are called after each keyboard scan without regard to whether a key was pressed, and after each call to NEXT. This background task appears both simultaneous with the current foreground task and instantaneous in reaction time.

25 Moore Chats on CompuServe
by Ward McFarland
One evening on CompuServe, Mr. Charles Moore fielded questions from others who were logged on to the information network at the time. The inventor of the Forth language had a good deal to say about Forth chips, Japanese technology and several other topics. This edited transcript may inspire you to plug in that modem and join the next live event!

DEPARTMENTS

5 Letters
6 Editorial: “Calls for Papers”
19 Advertisers Index
30 FIG Chapters
THEY’RE HERE!

at

SOFTWARE COMPOSERS

DELTA BOARD  4 MHz NC4000 Forth Engine CPU Board
MEMORY BOARD  128 KBytes — all CMOS
                Extended addressing of data memory
BACKPLANE     7 slots for system building and networking
                All I/O and memory port lines accessible

The Delta Evaluation System  $895
• 4 MHz Delta Board with Novix NC4000 Forth Chip on board.
• cmFORTH programming language interpreter and compiler in EPROM.
• User manual, board schematic, and user bulletin board support.
• 4K 16 bit words of static RAM and 4K words of EPROM.
• 8 selectable 256 word data stacks and return stacks for multi-tasking.
• 21 independently programmable single bit I/O ports.
• 4 1/2” x 6 1/2” board with 72-pin edge-connector bus with all major Novix signals.
• Delta Regulator Base with attached connector and single 5-volt wall mount power supply.
• Reset switch and serial port on board with RS232 connector and cable.
• 90 day warranty.
• Fully assembled, tested, and ready to use.

Additional Novix chips available for $195, 1-9 quantity.
Software Composers is an authorized Novix Distributor

For a copy of the Delta User’s Manual, send check for $35 (deductible from Delta Board purchases).

“I’m delighted to see Software Composers’ board on the market. It provides incredible capability and versatility with minimal parts, size and price. An excellent introduction to the new generation of hardware and software.”

Chuck Moore, November 1985

COMING SOON!

SCForth
An F83 development environment

THE DELTA DEVELOPMENT SYSTEM
A hardware/software/power supply/enclosure package

For product data and ordering information, write:

SOFTWARE COMPOSERS

Software Composers
210 California Avenue #F
Palo Alto, CA 94306
(415) 322-8763
Invisible, and Fairly Elegant:
32-Bit Forth

Dear Marlin,

Thanks for Volume VII! I particularly liked “Synonyms and Macros” and hot-patching. I had wanted these and hadn’t gotten around to writing them. They made Forth-83 look impressive: I could copy the 83-Standard ones word for word, and they worked before I understood them! The pseudo-interrupt technique is a nice idea; improved micros, suggestions.

I have not yet seen Forth implemented on a full thirty-two-bit machine. I am using Forth-83 on 64K of a 68000. I would like to see the thirty-two-bit standard look just like the sixteen-bit standard. DUP would be a thirty-two-bit DUP, 2DUP would be a sixty-four-bit DUP and, would allocate thirty-two bits in the dictionary, which would have a maximum size of roughly four gigabytes (or less, depending on the trouble and expense of excessive memory). Everything would seem quite familiar. Mr. Hore’s first point was that existing software should continue to run. Many simple Forth-83 programs would run without modification, except to a few basic words such as ARRAY. The problems that I imagine coming up are that, first, some things depend on going around the -32768/+32767 boundary and, second, sometimes people use ad hoc methods for things like arrays — for example, getting the address of an array element, and then adding two to get the address of the next element. Limiting a thirty-two-bit Forth to keep these functional would cripple it. Better for the thirty-two-bit Forth to be able to set aside 64K to run 79-Standard or 83-Standard programs.

Mr. Hore suggested setting up a constant named LSIZE which gives the stack word length, while the compilation address length stays at sixteen bits. @ and ! would transmit sixteen bits to allow DO ... @ ... 2+LOOP to work. So long as the manufacturers make LSIZE in byte multiples, we can set LSIZE to 2, 4, 6, etc. I personally have set up the synonym LS and I now attempt to remember to write DO ... @ ... LS + LOOP.

I see no problem with this general plan for thirty-two-bit Forth. If your program is written with LSIZE and you don’t use the -32768--32767 shunt (which I’ve used only for random number generators), it can be made to run in double precision with no run-time penalty but using twice the variable storage. If it might not be compatible, then you load a truncated vocabulary. Everything pretends to be sixteen bits starting at a base pointer, + and *, sign extend, etc. No problem.

What about segmented memory for sixteen-bit machines? I believe that this can be used very nicely within the standard. First, why not set aside 64K (more or less) for a nice editor and mass storage blocks? VOCABULARY can shift between 64K base addresses without the user having to pay any attention. If you had the memory, wouldn’t you like a nice, big editor and a little RAM disk that just waited for you without using any of your 64K?

Are you at all cramped in 64K? We can bend the standard just a little bit and put all the machine code behind a different base address. Again, EXECUTE picks the correct base address without your knowledge. Of course, if you want to modify your machine code from Forth, you will need a couple of special words. I and @ won’t do it any more.

Do you want big arrays? Why not let VARIABLE set up space in a new area? You can have up to 64K of variables with sixteen-bit addressing. If you set up VARIABLE1, which is just like VARIABLE except that it starts at a different base address, then you can have a 64K array. With a slightly more complicated definition for VARIABLE, you should be able to set up arrays with up to 64K per dimension, although you’ll run into physical limitations pretty fast.

The only reason Forth mixes machine code and Forth parameter values and variable contents together is for convenience, and with an 8086/8088 this isn’t really convenient any more. They can be split up in a way that is almost invisible and is fairly elegant.

I look forward to further discussion, preferably from people who have thought it out more than I have, but not so much more that I feel left out.

Sincerely,

J.E. Thomas
Birmingham, Alabama

Editor’s note: Reader Thomas and the author of the following letter will be glad to hear that we have scheduled Professor Yingve’s “Compiler Macros” as a follow-up to his “Synonyms and Macros” series.

A Synonym for fig-FORTH
Dear Marlin:

As a fig-FORTH user, I frequently find it necessary to convert Forth-79 and Forth-83 routines to fig-FORTH. Usually, the conversions are not too difficult, but occasionally they are. The difficulties nearly always arise from a lack of understanding of (Letters continued on page 6)
Calls for Papers

We have received several announcements of coming events — conferences and conventions so valuable in terms of technical knowledge imparted, personal and professional associations formed and perspective gained about Forth's place in the world, personal and professional associations formed and perspective that anyone who is serious about his understanding of Forth should attend at least one per year. As a long-time manager and attendee of both large and small meetings, I know the emotional “gearing up” and logistics required to spend a few days away from our normal routine. But I’ve found great interest and value in each of the formal Forth meetings of the past few years, and I think you will, too.

The annual Forth Conference held in Rochester, New York, is sponsored by the Institute for Applied Forth Research, Inc., in cooperation with IEEE and the University of Rochester. June 11–14 will mark the sixth such event, with topical emphasis on real-time artificial intelligence. Last year saw 175 attendees presenting more than sixty papers, participating in working groups and enjoying themselves at after-hours receptions and discussion groups. At this late date, the best way to get detailed information is to call the institute at 716-235-0168.

Casting our editorial net a bit further, we find that China is once again on the itinerary of Forth experts. (FIG members will remember our published account of the very valuable lecture tour there two years ago.) The notices we have received indicate an October 31 – November 2 “International Workshop on Forth and Its Applications” to be held at the National Taiwan Institute of Technology in Taipei, with optional, extended travel. Papers are being solicited, and must be received by September 30; abstracts are due earlier. For specific deadlines and a complete travel agenda, contact Dr. C.H. Ting at 415-424-3001, or evenings at 415-571-7639.

For the eighth consecutive year, the Forth Interest Group is hosting an Annual Forth National Convention. This hallmark event always features many illustrious speakers and the widest representation of Forth vendors anywhere. Hundreds of FIG members and the interested public will convene this year in Santa Clara, California on November 21–22 at the Doubletree Hotel. Preliminary plans show that this year’s meeting promises to be memorable; watch these pages for details of interest.

FORML Conferences are like dynamic, highly participatory graduate seminars on Forth implementation and programming techniques. Most attendees present advanced papers and space is limited, but some places are always available for those who choose only to immerse themselves in the advanced subject material and lively dialogue. The surroundings of California’s Asilomar Conference Center provide a rewarding background, located near Carmel on the scenic Monterey peninsula. November 28 – 30 are the scheduled dates, and more information will be published later in Forth Dimensions.

The personal and professional results gained from attending these events are many. In addition, they are a good source of feedback from members to the FIG leadership. We look forward to meeting many of you this year!

—Marlin Ouverson
Editor

(Letters, continued)

the way in which a word operates in Forth–83. Such was the case with SYNONYM, which appeared in Forth Dimensions VII/3.

SYNONYM is one of the better one-word tools I have seen. The difficulties with conversion to fig-FORTH arose out of the way in which the 83 FIND differs from -FIND, and the method of determining whether or not a word is immediate. I still don’t fully understand how those two operate in Forth–83, but I do know how they function in fig-FORTH.

Perhaps there are other fig-FORTH users who had the same problem. The attached listing contains the conversion to fig-FORTH.

Sincerely,

Gene Thomas
Little Rock, Arkansas

MacForum: World Wide . . . and Still Growing

Dear FIG:

I would like to make two announcements that may be of interest to your members.

First, for those using the Macintosh, there is now a National MacForth Users’ Group. We are a recently-formed, independent, not-for-profit organization with over 200 enthusiastic members world wide (and still growing). We have accumulated a library of over 5000 screens of public-domain MacForth source code on sixteen disks, which we distribute to our members at a nominal charge per disk. We have just published our first newsletter (twenty pages of dense text and code), and plan to continue this on about a bi-monthly basis. Many of our members have expressed interest in providing coverage of other Macintosh Forths in addition to MacForth, so if we get submissions in these areas, we will be glad to consider them for inclusion in our library or newsletter. Membership or newsletter subscription information can be obtained from NMFUG, 3081 Westville Station, New Haven, Connecticut 06515.

Secondly, the MacForth Forum on CompuServe has proved to be so successful in the past eighteen months that Creative Solutions, Inc. (the Forum sponsor and publisher of MacForth) and CompuServe have “gone public” with it. Although CSI is still taking care of the bureaucratic dealings with CompuServe, our SIG is no longer restricted to MacFortthers. We are now called the Forth Forum and are open to all CompuServe users, with no extra charge or special registration. We have ten areas for questions, messages, announcements, etc. (divided by topic), ten Data Library areas for files, articles, tutorials, etc., and a private Conference area usable at any time by members for any “live” group discussions. We have recently had Charles Moore in a two-hour open forum, and we plan to get other Forth “notables” on-line from time to time, in addition to frequent informal “chat sessions.” Our data libraries have inherited many MacForth files, but we are anxious to get lots of contributions of a more “standard” Forth nature from users of other systems and Forth dialects.

Happy Forthing,

Ward McFarland
New Haven, Connecticut

Grandfather’s DO LOOP

Dear Marlin,

Maybe it is uncouth of me to mention it, since Michael Hore intended it as a teaching example (Forth Dimensions VI/6, “Enhanced DO LOOP”), but there is a cleaner
implementation of his word `LOOKUP' that does not require any new control structures. The approach he illustrates initially I would characterize as the American (Australian?) approach to programming:

1. Of course there will be a match. Isn't that what we are looking in the table for?
2. Oops! In the wild possibility there isn't, we had better patch up the code to handle things.

In contrast, and in memory of my grandfather (b. Mild May, Ontario) and with the kind permission of the folks south of Detroit, I would like to offer what I would call the Canadian approach:

1. Things are very unlikely to get any better than they are right now.
2. If by some wild chance they do improve, grab the improvement and `LEAVE' cleanly.

The key to simplifying Mr. Hore's LOOKUP is to realize that we can put the default zero (standing for "not found") on the stack before we enter the loop — and therefore have to put it there only once. If we find a match, we `replace' the default value with the right address. Then, all we have to do is arrange it so that each pass through the loop starts with `value' and `addr' both on the stack, with `addr' possibly subject to change if we haven't hit the end of the loop yet. Then, no matter how we leave the loop, `value' and `addr' will be on the stack when we have left. Then, since we no longer need `value', we can `DROP' it. Thus, we have a simple example of the loop invariant so beloved by computer scientists.

`addr' always represents the true current state of affairs, which is "not found" until we find something, at which point `addr' changes.

The complete source code is found on the screen.

**HOLD for Prettier Numbers**

In the article "Making Numbers Pretty" (VII/5), a last-minute enhancement was untested and proved wrong. In the word `16BITS', the output is properly spaced by using `32 HOLD' (32 being the ASCII value for a blank). In the word as presented, `SPACE' and `SPACES' are executed during the conversion of the number to an ASCII string. By the time the conversion is complete and the string is ready for `TYPE', all the spaces have already been displayed. That was not the idea. The spaces were to be interspersed among the numeric characters, and that is achieved through `HOLD.'

—Michael Ham

---

**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
Interrupt-Driven Serial Input

John S. James
Santa Cruz, California

Difficulties of implementing interrupt-driven serial I/O have caused a bottleneck in programming for PC compatibles. Input is especially important, because otherwise characters can be lost during screen scroll at speeds above 300 bps, or lost when other tasks are running simultaneously.

By noting just how the difficulty arises, we can learn to avoid such problems in other contexts.

Here, the problems stem from lack of documentation, especially examples. It's hard to find all of the necessary information in one place. Lack of examples causes special problems in cases like this, where there is no environment for experimentation — where everything must be right before anything happens.

This article provides a working example of serial input (screens 2 – 4). The other screens include file I/O (the “new” DOS calls), and a simple terminal program which illustrates use of the interrupt routines.

This example is not a finished product. It could use a number of improvements, such as saving and restoring the interrupt vectors; we didn’t add that here because there wasn’t time for thorough testing before this article went to press. This program uses the character input buffer to save a copy of the session for writing to disk — good enough for a test of interrupts, but a better design would use a circular input buffer, and probably a second task to manage the save buffer. Here we aimed for simplicity and compatibility in illustrating use of the interrupt words.

Screen 2

We placed the interrupt service routine first, to emphasize that it doesn’t depend on the other parts of the program.

One helpful trick in getting up a complex code word such as this one: if there's any question of whether the assembly is working as intended, use the disassembler in the DEBU< command provided with the operating system. For example, if you are running F83, use DEBU< F83.COM; then use the G (Go) command to start running Forth. Get the hex address where the code begins (usually two bytes above the address returned by “tick” in Forth-83), then use CONTROL-C to get out of Forth and back to the debugger. Then use the U (Unassemble) command with the address of the code.

Screen 3

The word DOS2 is a handy call for many of the “new” MS-DOS 2.0+ I/O calls. Here we also use it to set the system interrupt vector to point to our service routine. ISETUP also sets some necessary control bits. (PCT, not a standard word but available in most systems, writes a byte to a port).

Screen 4

A mask turns the COM1 interrupt on or off. It’s important to stop the interrupts whenever accessing PTR, the variable used by the interrupt routines. A “productized” interrupt system would hide these internals and only allow access through a defined set of calls.

Screen 5

If there are any characters in the buffer, MREAD returns the next one; otherwise it returns a -1 to indicate that the buffer is empty.

ION-OFF probably aren’t necessary here, since MREAD is being used on a sixteen-bit architecture. On an eight-bit machine, an interrupt (and increment of PTR) could occur between the fetches of the two bytes unless interrupts were turned off. Such bugs can hide for a long time, then cause problems which are hard to track down.

An improved interrupt routine, for a general-purpose software library, would hide these problems so that users of the routine would not need to be concerned about them.

Screen 6

The (non-interrupt) output in MWRITE got a little complicated to prevent problems with certain equipment.

Some external modems do not handshake properly if the BIOS serial-I/O call is used. It may be impossible to send anything, even commands, when the modem first comes up. Some modems have a switch to override this problem, other do not; so it’s better to use direct output to the port and not be so fussy about handshaking.

Without line 12, some modems will drop the line and report “no carrier” immediately, just after the phone call has gone through.

Screens 7 – 10

We define enough of the “new” DOS calls for our purposes. For more information about these calls, see the Disk Operating System Technical Reference manual, or see any of a number of books on assembly-language programming of the PC.

FSTART uses these DOS calls to open a file if it exists, or to create it otherwise. FDO opens or creates a file, writes the buffer which has saved a record of the terminal session, then closes the file. The file remains closed almost all the time, so that it will be protected in case of power loss or other abnormal termination of the session.

Screen 9 provides a default file name and lets the user change it. Screen 10 defines some miscellaneous words, which are mostly self-explanatory. (The definition of AT prevents accidental crashes during testing, if the AT command intended for a Hayes-compatible modem is mistakenly typed into F83 instead.)

Screen 11

The buffer is cleared (by KILL-FILE) whenever it is Saved to disk, so the user can do a SAVE as often as necessary, concatenating the new data to whatever may have been on the disk before.

This system is always saving the complete terminal session in the buffer. A common use of the Kill command is to delete the record of the logon, which may include a password.

Users should stop the remote system during a Save, or characters can be lost, with the implementation given here. The automatic save — seldom necessary in attended operation, as the user should Save before the large buffer has filled — will lose characters on the screen, but not the file, unless the remote system fails to respond to XOFF-XON. A more sophisticated design could correct these problems.

Screens 12 and 13

ESCAPE-FN tests any command from the terminal user, and calls the proper function.

The main word TERM is largely self-explanatory.
Screen 14

This optional screen is not loaded by screen 1, because it applies only to F83, and it is used only after development is complete. Loading this screen causes TERM to execute automatically when the COM file containing the object program is loaded.

For practical use of this program, note that there are two ways to lose data, ways you may want to remove. The program has an undocumented escape exit into Forth, which you can easily remove on Screen 12. Also, a control-C will abort to the operating system, in many Forths. You may want to use a system call which doesn’t check for control-C in some situations.

User Interface Note

Designers should consider the simple user interface illustrated in this program, using one-keystroke, usually one-line menus.

When the program starts, it identifies the one key needed for help (here, Escape). This help key also gives the user control, often through the top-level menu (which is the only menu, in this example).

Beginner and expert modes are the same. Beginners get the prompts they need, for all their options. Experts can type the same keystrokes at full typing speed, then perhaps glance at the screen to see that everything worked as planned. Every selection is a single keystroke, unless a name, etc., is required, in which case the Enter key terminates it.

Although this menu system is also the Help system, additional explanation, if needed, can fit under options labeled (H)elp or (?).

Developers can implement tree structures of these menus easily, either by hard coding or by a table-driven system.

Either function keys or alphabetic commands can be used. Function keys have the advantage for users who are not touch typists; for typists, alphabetic commands are easier to find on the keyboard, and they are more mnemonic than function keys. This menu system usually uses short, one-line menus, so there are plenty of keys on the keyboard. (Compare with conventional function-key systems which often load forty commands onto ten keys, requiring multiple keystrokes, templates and arbitrary grouping of commands.)

Note that most user errors will have no effect in this system. Any non-valid selection will terminate the menu operation, and do nothing else. An erroneous selection sequence cannot do damage unless it reaches a “leaf” of the menu tree, meaning that each character typed happened to match an existing choice of the menu then in effect. And of course the dangerous commands can ask for confirmation, for additional protection.

This user interface also gives the system designer the flexibility to group the choices in a rational way. Users who do not use an item need never see it.

```
Scr # 2  A:FTERM.BLK
0 ( Interrupt service routine  4-10-86 )
1 HEX
2 VARIABLE PTR ( Where interrupt routine will put characters )
3 3FD CONSTANT LSTATUS ( Line status register )
4 3FA CONSTANT RBUFFER ( Receive buffer )
5 ASSEMBLER
6 CREATE IIN ( Interrupt service routine, Port 1, Input )
7 AX PUSH BX PUSH DX PUSH ( Save registers )
8 LSTATUS # DX MOV DX AL IN 01 # AL TEST ( Data ready? )
9 0<> IF RBUFFER # DX MOV DX AL IN ( Read character )
10 CS: PTR # BX MOV ( Move it to (PTR) )
11 CS: AL O (BX) MOV BX INC ( Inc PTR )
12 CS: BX PTR #) MOV THEN
13 DX POP BX POP ( Restore these registers )
14 20 # AL MOV 20 # AL OUT ( 8259A Interrupt controller )
15 AX POP ( Restore ) IRET FORTH DECIMAL
```

FORTHkit
Assemble a 4 Mips Computer!

PARTS
4MHz Novix NC4000
4x6" mother-board
Press-fit sockets
2 4kx8 PROMs

INSTRUCTIONS
cmFORTH listing
Application Notes
Brodie on NC4000

ASSEMBLY
Buy 6 RAMs
Misc. parts
Press 360 sockets
Soldeer 3 capacitors
2 resistors
Attach 200mA @ 5V
RS-232 cable
Insert 11 chips

Program host as terminal/disk
(1 screen of Forth)

LEARN
Modern technology
Interface design

EXPLORE
High-speed Forth
On-board interfaces
• 16-bit parallel
• video
• floppy
• printer
Plug-in interfaces
4 pin/socket busses
Battery power (6V)

$400
Inquire for details

Chuck Moore
COMPUTER COWBOYS
410 Star Hill Road
Woodside, CA 94602
(415) 851-4362
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 resume included in our database, or if you are looking for a FORTH Programmer, contact us or send your resume to:

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

Committed to Excellence
Scr # 7  AFTERM.BLK
0 ( File I/O 4-10-86 )
 1 HEX
 2 : FCREATE ( nattrib ascizi -- nhandle )
 3 0 ROT 3000 DOS2 ;
 4 : FOPEN ( nacces ascizi -- nhandle )
 5 SWAP 0 0 ROT 3000 + DOS2 ;
 6 : FCLOSE ( nhandle -- nread )
 7 0 0 ROT 3E00 DOS2 ;
 8 : FREAD ( ndata ndata nhandle -- f )
 9 3F00 DOS2 ;
10 : FWRITE ( ndata nhandle -- swrittten )
11 4000 DOS2 ;
12 ( Note - should test case flag - make error arg negative )
13 : FMOVE ( dbytes method nhandle -- f Move file pointer )
14 SWAP 4200 + DOS2 ;
15 DECIMAL

Scr # 8  AFTERM.BLK
0 ( Files - write bytes to new file, or concat. to old 4-10-86 )
 1 2 CONSTANT READ-WRITE ( Acces code: 0=read, 1=write, 2=both )
 2 2 CONSTANT FROM-EOF ( Method code to move path )
 3 0 = offset from beginning of file, 1 = from current location,
 4 2 = end of file - offset )
 5 : FSTART ( ascizi -- nhandle Open or create a file )
 6 READ-WRITE OVER FOPEN
 7 DUP -2 = IF ( Not found ) DROP 0 SWAP FCREATE
 8 ELSE SWAP DROP DPU 0 0 ROT FROM-EOF SWAP FMOVE DROP
 9 THEN ;
10 : FDO ( ad nd ascizi -- Write or open concatenated )
11 FSTART DUP >R FWRITE DROP R> FCLOSE DROP ;
12
13
14
15

Scr # 9  AFTERM.BLK
0 ( Files - Default name, and word to change the name 4-10-86 )
 1 CREATE FILE-NAME 66 C, 58 C, 84 C, 69 C, 62 C, 77 C, 0 C,
 2 74 ALLOT ( Default name is SiTERM - ASCII format )
 3 : GET-STRING ( a = ASCII input )
 4 DUP 80 EXPECT
 5 SPAN @ = = IF SPAN @ = 0 SWAP C! ELSE DROP THEN ;
 6 : GET-NAME ( -- )
 7 CR ." Name now is: " FILE-NAME @O OVER + SWAP DO
 8 I @O = = IF LEAVE ELSE I @O EMI Then LOOP
 9 CR ." New name: " FILE-NAME GET-STRING CR ;
10
11
12
13
14
15

Scr # 10  AFTERM.BLK
0 ( Terminal program - miscellaneous words 4-10-86 )
 1 VARIABLE FIRST-TEST TRUE FIRST-TEST ;
 2 : FIRST-TIME ( -- F ) FIRST-TEST @ FALSE FIRST-TEST ;
 3 : ESC-MSG ( -- )
 4 CR ." (SAVE (NAME) (K)ill (U)kill (Z)exit" CR ;
 5 AT ." WOOPS! " ( Avoid crash from easy mistake )
 6 VARIABLE BPTR ( Backup value of point - for UNKILL )
 7 VARIABLE BLPTR ( Backup last value of pointer )
 8 : TSAVE PAD ; ( Beginning of terminal buffer )
 9 : TSAVE-END FIRST ; ( End of terminal buffer )
10
11
12
13
14
15

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 Fort with all currently defined words onto disk as a .COM file.

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

Source code is available as an option.

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 11-26 volts at 12ma, plus current required for options. $240 - $360

Up to 24kb memory: 2kb 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 buse available at optional header to allow expanded capability via user provided interface board.

Micro Computer Applications Ltd
8 Newfield Lane
Newtown, CT 06470
203-426-8164

Foreign orders add 5% shipping and handling. Connecticut residents add sales tax.
CALL FOR PAPERS
for the eighth annual
FORML CONFERENCE

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

Following Thanksgiving
November 28 - 30, 1986

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

Theme: Extending Forth towards the 87-Standard

The Forth Standards Team is expected to hold sessions in 1987 to consider proposals for
updating the Forth standard. FORML isn't the place to standardize Forth, but it is the forum
to present and discuss your ideas. Papers are invited that address relevant issues such as:

Portability of applications, decompilers & debuggers.
Hardware Forth processors.
Large address spaces like 32-bit computers.
Control structures, data structures & strings.
Files, graphics, & floating point operations.
Cohabitation with operating systems, other languages & networks.

Papers on other Forth topics are also welcome. Mail your abstract(s) of 100 words or less by
September 1, 1986 to:

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

Completed papers are due October 1, 1986. For registration information call the Forth
Interest Group business office at (408) 277-0668 or write to FORML

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.
AFTERM.BLK
Terminal program - write to file, and autosave 4-10-86

WRITE-FILE ( -- )
  IOFF TSAVE PTR @ TSAVE - FILE-NAME ION FDO ;
  ( Interrupts may be left on during actual write to the file,
  but as this program is now written, transmission should
  stop before file I/O, because input buffer is cleared
  after the write. )
  KILL-FILE ( -- )
  IOFF TSAVE PTR @ PTR TSAVE ...
  IF ( Not empty already )
  PTR @ BPTR ! LPTR @ BLPTR ! TSAVE PTR @ TSAVE LPTR ! THEN ION ;
  ?AUTOSAVE ( -- If buffer nearly full, save automatically )
  IOFF PTR @ PTR ION FDO ;
  ( Interrupts may be left on during actual write to the file,
  but as this program is now written, transmission should
  stop before file I/O, because input buffer is cleared
  after the write. )
  WRITE-FILE KILL-FILE
  MWRITE XOFF 10000 0 DO LOOP ( Delay )

TERM ( -- Dumb terminal program )
  CR . "Use ESC for help" CR
  FIRST-TIME? IF 1200 MINITIALIZE TSAVE PTR @ TSAVE LPTR ! THEN
  ISETUP ION THEN
  FALSE ( Loop control - set to TRUE to exit )
  BEGIN
    ?AUTOSAVE
    MREAD ( Read one character )
    DUP 1 - IF DROP ELSE EXIT THEN
    KEY IF ( If key typed, rend it, unless Esc )
    KEY DUP 27 = IF DROP ESC-MSG ( Escape key )
    KEY DUP 90 > IF 32 - THEN ( Make it upper case )
    ESCAPE-FN IF DROP 1 THEN
    ELSE MWRITE THEN
    ELSE MWRITE THEN
    THEN
  DUP UNTIL DROP ;

HELLO2 only - sets automatic execution
HELLO2 ONLY FORTH ALSO DEFINITIONS TERM ;
HELLO2 IS BOOT

SAVE-SYSTEM TERM.COM

FORTH
The computer language for increased...
EFFICIENCY
reduced.....
MEMORY
higher......
SPEED
MVP-FORTH SOFTWARE
Stable...Transportable...
Public Domain...Tools
MVP-FORTH PROGRAMMER'S KIT
for IBM, Apple, CP/M, MS/DOS, Amiga, Macintosh
and others. Specify computer. $175

MVP-FORTH PADS,
a Professional Application Development System. Specify computer.
$500

MVP-FORTH EXPERT-2 SYSTEM
for learning and developing knowledge based programs.
$100

Word/Kalc,
a word processor and calculator system for IBM.
$150

Largest selection of FORTH books: manuals, source listings,
software, development systems and expert systems.
Credit Card Order Number: 800-321-4103
(In California 800-468-4103)
Send for your FREE FORTH CATALOG

MOUNTAIN VIEW PRESS
PO BOX 4656
Mountain View, CA 94040

Volume VIII, No. 1 13
Fast Fixed-Point Trig

Johann Borenstein
Haifa, Israel

Based on J. Baumgarner’s article “Fixed-Point Trig by Derivation,” Forth Dimensions IV/1, I have written a modified version for the sine function. While this version of \texttt{SIN} is approximately as long (in terms of compiled code) and almost as accurate as the original definition, it is about nine times faster (5.3 msec on my Z80A system running at 3.75 MHz). This considerable increase in speed has been achieved by optimal scaling of the series parameters, to the extent that no divisions are performed to evaluate the series. Also, the improved \texttt{SIN} is strictly in Forth.

To see how the modified version works, let’s start with the basic Taylor-Maclaurin series expansion for the sine function as in (1).

By successively factoring out \(x\) and \(x^2\), the series can be written as in (2).

When using a scaled integer \(x\), each multiplication must be divided by the scaling factor.

\begin{align*}
(1) \quad \sin x &= x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \ldots \\
(2) \quad \sin x &\approx x \left(1 - \frac{x^2}{6} \left(1 - \frac{x^2}{20} \left(1 - \frac{x^2}{42} \left(1 - \frac{x^2}{72} \right)\right)\right)\right) \\
(3) \quad \sin x &\approx \frac{y}{k} \left(1 - \frac{y^2}{6k^2} \left(1 - \frac{y^2}{20k^2} \left(1 - \frac{y^2}{42k^2} \left(1 - \frac{y^2}{72k} \right)\right)\right)\right) \\
(4) \quad z &= \frac{K_1}{K_2} x \quad ; \quad \frac{K_1}{K_2} >> 1 \\
(5) \quad \sin x &\approx \frac{K_2}{K_1} z \left(1 - \left(\frac{K_2}{K_1}\right)^2 \frac{z^2}{6} \left(1 - \left(\frac{K_2}{K_1}\right) \frac{z^2}{20} \left(1 - \left(\frac{K_2}{K_1}\right) \frac{z^2}{42} \left(1 - \frac{K_2}{K_1} \frac{z^2}{72} \right)\right)\right)\right) \\
(6) \quad xs &= \frac{z \times z}{k_1} \\
(7) \quad \frac{K_1}{K_2} \sin x &= \frac{z}{K_1} \left(\frac{K_2}{6} \cdot \frac{xs}{k_1} \left(\frac{K_2}{20} \cdot \frac{xs}{k_1} \left(\frac{K_2}{42} \cdot \frac{xs}{k_1} \left(\frac{K_2}{72} \cdot \frac{xs}{k_1} \right)\right)\right)\right) \\
(8) \quad 3784 \sin x &= \frac{z}{2^{16}} \left(2^{16} - \frac{50 xs}{2^{16}} \left(2^{16} - \frac{15 xs}{2^{16}} \left(2^{16} - \frac{7.14 xs}{2^{16}} \left(2^{16} - 4.2 \cdot xs \right)\right)\right)\right) \\
(9) \quad \text{scaling factor} &= \frac{1}{K_1/K_2} = \frac{1}{3784} = 0.026 \% \\
(10) \quad a \ b -- m \text{ with } m = 2^{16} - \frac{a \times x^2 \times b}{2^{16}}
\end{align*}

(Continued on page 19)
MEMBERSHIP
IN THE FORTH INTEREST GROUP

108 - MEMBERSHIP in the FORTH INTEREST GROUP & Volume 8 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 worldwide 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 publication discounts, 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 may select surface ($37.00) or air ($43.00) delivery.

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 which entitles you to a 10% discount on publications from FIG. Your member number will be required to receive the discount, so keep it handy.

HOW TO USE THIS FORM
1. Each item you wish to order lists three different Price categories:

   Column 1 - USA, Canada, Mexico
   Column 2 - Foreign Surface Mail
   Column 3 - Foreign 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 - Volume 1 FORTH Dimensions (1979/80) $15/16/18
102 - Volume 2 FORTH Dimensions (1980/81) $15/16/18
103 - Volume 3 FORTH Dimensions (1981/82) $15/16/18
104 - Volume 4 FORTH Dimensions (1982/83) $15/16/18
105 - Volume 5 FORTH Dimensions (1983/84) $15/16/18
106 - Volume 6 FORTH Dimensions (1984/85) $15/16/18
107 - Volume 7 FORTH Dimensions (1985/86) $20/21/24

ASSEMBLY LANGUAGE SOURCE CODE LISTINGS
Assembly Language Source Listings of fig-Forth for specific CPUs and machines with compiler security and variable length names.

513 - 1802/MARCH 81 $15/16/18
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
520 - ALPHA MICRO/SEPT 80 $15/16/18
521 - APPLE II/AUG 81 $15/16/18
522 - ECLIPSE/OCT 82 $15/16/18
523 - IBM-PC/MARCH 84 $15/16/18
524 - NOVA/MAY 81 $15/16/18
525 - PACE/MAY 79 $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
### Books about Forth

<table>
<thead>
<tr>
<th>Title</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL ABOUT FORTH</td>
<td>$25/26/35</td>
</tr>
<tr>
<td>BEGINNING FORTH</td>
<td>$17/18/21</td>
</tr>
<tr>
<td>COMPLETE FORTH</td>
<td>$16/17/20</td>
</tr>
<tr>
<td>FORTH ENCYCLOPEDIA</td>
<td>$25/26/35</td>
</tr>
<tr>
<td>FORTH FUNDAMENTALS, V. 1</td>
<td>$16/17/20</td>
</tr>
<tr>
<td>FORTH FUNDAMENTALS, V. 2</td>
<td>$13/14/16</td>
</tr>
<tr>
<td>FORTH NOTEBOOK</td>
<td>$25/26/35</td>
</tr>
<tr>
<td>INSIDE F83</td>
<td>$25/26/35</td>
</tr>
<tr>
<td>LEARNING FORTH</td>
<td>$17/18/21</td>
</tr>
<tr>
<td>MASTERING FORTH</td>
<td>$18/19/22</td>
</tr>
<tr>
<td>STARTING FORTH (soft cover)</td>
<td>$20/21/24</td>
</tr>
<tr>
<td>STARTING FORTH (hard cover)</td>
<td>$24/25/29</td>
</tr>
<tr>
<td>THINKING FORTH (soft cover)</td>
<td>$16/17/20</td>
</tr>
<tr>
<td>THREADED INTERPRETIVE LANGUAGES</td>
<td>$23/25/28</td>
</tr>
<tr>
<td>UNDERSTANDING FORTH</td>
<td>$3.50/5/6</td>
</tr>
</tbody>
</table>

- **Glen B. Haydon**
  - An annotated glossary for MVP Forth; a 79-Standard Forth.

- **Paul Chirlian**
  - Introductory text for 79-Standard.

- **Alan Winfield**
  - A comprehensive introduction including problems with answers. (Forth 79)

- **Mitch Derick & Linda Baker**
  - A detailed look at each fig-Forth instruction.

- **Kevin McCabe**
  - A textbook approach to 79-Standard Forth.

- **Dr. C. H. Ting**
  - A glossary.

- **Gary Feierbach & Paul Thomas**
  - The standard tools required to create and debug Forth-based applications.

- **Dr. C. H. Ting**
  - Invaluable for those using F-83.

- **Margaret A. Armstrong**
  - Interactive text, introduction to the basic concepts of Forth. Includes section on how to teach children Forth.

- **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.

- **Leo Brodie**
  - A lively and highly readable introduction with exercises.

- **Leo Brodie**
  - The sequel to “Starting Forth”. An intermediate text on style and form.

- **R.G. Loeliger**
  - Step-by-step development of a non-standard Z-80 Forth.

- **Joseph Reymann**
  - A brief introduction to Forth and overview of its structure.

### Forml Conference Proceedings

FORML 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.

<table>
<thead>
<tr>
<th>Title</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORML PROCEEDINGS 1980</td>
<td>$30/33/40</td>
</tr>
<tr>
<td>FORML PROCEEDINGS 1981 (2V)</td>
<td>$45/48/50</td>
</tr>
<tr>
<td>FORML PROCEEDINGS 1982</td>
<td>$30/33/40</td>
</tr>
<tr>
<td>FORML PROCEEDINGS 1983</td>
<td>$30/33/40</td>
</tr>
<tr>
<td>FORML PROCEEDINGS 1984</td>
<td>$30/33/40</td>
</tr>
<tr>
<td>FORML PROCEEDINGS 1985</td>
<td>$35/38/45</td>
</tr>
</tbody>
</table>

- **FORML Conference Proceedings**
  - Technical papers on the Forth language and extensions.
  - Nucleus layer, interactive layer, extensible layer, metacomilation, system development, file systems, other languages, other operating systems, applications and abstracts without papers.
  - Forth machine topics, implementation topics, vectored execution, system development, file systems and languages, applications.
  - Forth in hardware, Forth implementations, future strategy, programming techniques, arithmetic & floating point, file systems, coding conventions, functional programming, applications.
  - 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.

### 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.

<table>
<thead>
<tr>
<th>Title</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROCHESTER 1981 (Standards Conference)</td>
<td>$25/28/35</td>
</tr>
<tr>
<td>ROCHESTER 1982 (Data bases &amp; Process Control)</td>
<td>$25/28/35</td>
</tr>
<tr>
<td>ROCHESTER 1983 (Forth Applications)</td>
<td>$25/28/35</td>
</tr>
<tr>
<td>ROCHESTER 1984 (Forth Applications)</td>
<td>$25/28/35</td>
</tr>
<tr>
<td>ROCHESTER 1985 (Software Management and Engineering)</td>
<td>$20/21/24</td>
</tr>
</tbody>
</table>

- **ROCHESTER**
  - 79-Standard, implementing Forth, data structures, vocabularies, applications and working group reports.
  - Machine independence, project management, data structures, mathematics and working group reports.
  - 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.
  - Forth in image analysis, operating systems, Forth chips, functional programming, real-time applications, cross-compilation, multi-tasking, new techniques and working group reports.
  - 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 #1 $15/16/18
Robotics.

402 - JOURNAL OF FORTH RESEARCH V.1 #2 $15/16/18
Data Structures.

403 - JOURNAL OF FORTH RESEARCH V.2 #1 $15/16/18
Forth Machines.

404 - JOURNAL OF FORTH RESEARCH V.2 #2 $15/16/18
Real-Time Systems.

405 - JOURNAL OF FORTH RESEARCH V.2 #3 $15/16/18
Enhancing Forth.

406 - JOURNAL OF FORTH RESEARCH V.2 #4 $15/16/18
Extended Addressing.

407 - JOURNAL OF FORTH RESEARCH V.3 #1 $15/16/18
Forth-based laboratory systems and data structures.

REPRINTS
420 - BYTE REPRINTS $5/6/7
Eleven Forth articles and letters to the editor that have appeared in Byte magazine.

421 - POPULAR COMPUTING 9/83 $5/6/7
Special issue on various computer languages, with an in-depth article on Forth's history and evolution.

DR. DOBB'S JOURNAL
This magazine produces an annual special Forth issue which includes source-code listings 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 listings.

503 - USING FORTH $20/21/23
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.

316 - BIBLIOGRAPHY OF FORTH REFERENCES $15/16/18
An excellent source of references to articles about Forth throughout microcomputer literature. Over 1300 references.

MISCELLANEOUS
601 - T-SHIRT SIZE
Small, Medium, Large and Extra-Large. White design on a dark blue shirt. $10/11/12

602 - POSTER (BYTE Cover) $15/16/18

616 - HANDBY REFERENCE CARD FREE

683 - FORTH-83 HANDBY 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 5¼' disks, they are IBM MSDOS 2.0 and up compatible. The disks are compatible with Forth-83 systems currently available from several Forth vendors. Macintosh 3½' disks are available for MasterFORTH systems only.

Forth-83 Compatibility IBM MSDOS
Laxen/Perry F83 LMI 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 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. These 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.

Please specify disk size when ordering.
## FORTH INTEREST GROUP

**P.O. BOX 8231**  
**SAN JOSE, CALIFORNIA 95155**  
**408/277-0668**

<table>
<thead>
<tr>
<th><strong>Name</strong></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Member Number</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Company</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Address</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>City</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>State/Prov.</strong></td>
<td><strong>ZIP</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Country</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Phone</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**OFFICE USE ONLY**

<table>
<thead>
<tr>
<th><strong>By</strong></th>
<th><strong>Date</strong></th>
<th><strong>Type</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Shipped By</strong></td>
<td><strong>Date</strong></td>
<td></td>
</tr>
<tr>
<td><strong>UPC Wt.</strong></td>
<td><strong>Amt.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>USPS Wt.</strong></td>
<td><strong>Amt.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>BO Date</strong></td>
<td><strong>Wt.</strong></td>
<td><strong>Amt.</strong></td>
</tr>
<tr>
<td><strong>By</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>ITEM #</strong></th>
<th><strong>TITLE</strong></th>
<th><strong>AUTHOR</strong></th>
<th><strong>QTY</strong></th>
<th><strong>UNIT PRICE</strong></th>
<th><strong>TOTAL</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>108</td>
<td>MEMBERSHIP</td>
<td></td>
<td></td>
<td></td>
<td>SEE BELOW</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- □ Check enclosed (payable to: **FORTH INTEREST GROUP**)
- □ VISA  
  □ MASTERCARD

<table>
<thead>
<tr>
<th><strong>Card #</strong></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Expiration Date</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Signature</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**(15.00 minimum on charge orders)**

## PAYMENT MUST ACCOMPANY ALL ORDERS

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

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

**PRICES**  
All orders must be prepaid. Prices are subject to change without notice. Credit card orders will be sent and billed at current prices. A $15 minimum on charge orders. Checks must be in U.S. drawn on a U.S. 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**  
Delivered to Alameda, Contra Costa, San Mateo, Los Angeles, Santa Cruz Counties, add 6 1/2%; Santa Clara County, add 7%; other California Counties, add 6%.

**VOLUME VIII, NO. 1**  
**30/37/43**  
**MEMBERSHIP FEE**

<table>
<thead>
<tr>
<th><strong>SUBTOTAL</strong></th>
<th><strong>10% MEMBER DISCOUNT</strong></th>
<th><strong>MEMBER #</strong></th>
<th><strong>SUBTOTAL</strong></th>
<th><strong>CA RESIDENTS SALES TAX</strong></th>
<th><strong>HANDLING FEE</strong></th>
<th><strong>TOTAL</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$2.00</td>
<td></td>
</tr>
</tbody>
</table>

**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**  
Delivered to Alameda, Contra Costa, San Mateo, Los Angeles, Santa Cruz Counties, add 6 1/2%; Santa Clara County, add 7%; other California Counties, add 6%.

**VOLUME VIII, NO. 1**  
**30/37/43**  
**MEMBERSHIP FEE**

<table>
<thead>
<tr>
<th><strong>SUBTOTAL</strong></th>
<th><strong>10% MEMBER DISCOUNT</strong></th>
<th><strong>MEMBER #</strong></th>
<th><strong>SUBTOTAL</strong></th>
<th><strong>CA RESIDENTS SALES TAX</strong></th>
<th><strong>HANDLING FEE</strong></th>
<th><strong>TOTAL</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$2.00</td>
<td></td>
</tr>
</tbody>
</table>

**VOLUME VIII, NO. 1**  
**30/37/43**  
**MEMBERSHIP FEE**
Defining a variable (6) as the repeatedly used square term, we can rewrite (5) as in (7).

As can be seen, there are five divisions by \( k_1 \), equations (6) and (7), which are rather time-consuming. In Forth there is a way that allows for extremely fast division by \( 2^{16} \). Using assembly language, this corresponds to sixteen right shifts, whereas in Forth the simple DROP does the job. Therefore, a scaled multiplication with \( 2^{16} \) as the scaling factor may be coded as:

\[
\begin{align*}
&z \ z \ u \ \text{SWAP} \ \text{DROP} \\
&\text{which is the same as} \\
&z \ z \ k_1 \ / \ \text{with} \ k_1 = 2^{16} \ \text{but much faster.}
\end{align*}
\]

More time is saved when the division by the series factors (6, 20, 42, 72) is replaced by multiplications, as may be done by appropriate choice of \( k_2 \).

Here \( k_2 \) has been chosen as \((k_2)^2 = 300\) such that the series finally becomes as in (8) where \( 3784 = k_1 / k_2 = 2^{16}/\sqrt{300} \)

A little precision has been sacrificed (for the sake of speed) by using the integers 7 and 4 instead of the factors 7.14 and 4.2 in (8). The average precision for the series is now 0.06%, and there is no point in trying to increase precision since it is in any case limited by the scaling factor according to (9).

Description of the Source Screens

Screen 1 holds the basic definition \textbf{SIN1} which evaluates the sine of values between zero and 5994 (3784 * P1/2 = 5944), corresponding to zero and 90°. The series may only be evaluated for arguments greater than 256 (= 4°), since the scaled square of anything smaller than 256 is less than one (therefore zero for integers) and corrupts the series. Fortunately, the sine of very small angles is almost equal to the angle itself, so that argument itself may be used as the result. For the worst case (argument = 256) this simplification yields an error of 0.08% which is only little more than the average error for the whole series. The term \[
\frac{z \times z}{2^{16}}
\]

is calculated and stored as \( XS \). Then the innermost bracket \( 2^{16} - 4 \) \( * \) \( XS \) is calculated. \textbf{TERM1} is called for the remaining elements, where \textbf{TERM1} evaluates the frequently used expression (10).

The accumulated \textbf{TERMS} are multiplied by \( z/2^{16} \) to obtain the scaled result. \textbf{DEG} scales whole-degree angles to the input range required by all the trigonometric functions \( f(z) \). \textbf{KTIMES} operates on the result of all trigonometric calculations and scales it to 1000 \( * \) \( f(z) \).

\textbf{DEG} and \textbf{KTIMES} are used for debugging only. They should not be used in a working application. Example:

Screen 2 holds definitions of additional trigonometric functions, all based on \textbf{SIN1}. \textbf{SIN1, COS1} and \textbf{TAN1} are about twice as fast as \textbf{SIN, COS} and \textbf{TAN} but accept input only in the range of \( 0 < z < 5944 \) (angles between 0 and 90°), whereas \textbf{SIN, COS} and \textbf{TAN} accept any input between \(-2^{15} \) and \(+2^{15} - 1\).

30 \textbf{DEG} \textbf{SIN1} \textbf{KTIMES} = 500
90 \textbf{DEG} \textbf{SIN1} \textbf{KTIMES} = 1000

Due to space limitations, Mr. Bornstein's code will appear in the next issue — Ed.
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 Metacomplier
- 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, 2.80, 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.

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, 7651-1865
UK: System Science Ltd., London, 01-248 0982
France: Micro-Sigma S.A.R.L., Paris, (1) 42.85.95.16
Japan: Southern Pacific Ltd., Yokohama, 045-314-9514

FORTH Dimensions 20
Volume VIII, No. 1
Case Conversion in KEY

David W. Harralson
Yorba Linda, California

The fig-FORTH definition of KEY returns the raw value of a keystroke. Most keyboards return the lower-case value of the pressed key unless shift-lock is pressed, and on older terminals you cannot then enter numbers without taking off the shift-lock.

Since Forth words are usually in upper case, it would be desirable to have Forth perform the lower-to-upper translation automatically, with provisions to disable case translation, instead of having translation words you call after parsing a character string.

Usually, you want lower-to-upper translation. However, when inputting prompts or comments, you may want lower-case input without having to manually reset case conversion. For instance, if you are in an editor, you don’t want to exit the editor and reset case translation just to enter a lower-case string. In this case, you would want the key case to be inverted, lower-case keys being translated to upper case, and vice versa.

I have implemented this capability by adding a case-translating word (0 = no translation, 1 = translate to upper case, 2 = switch cases), and words to set which case translation you want (LC provides no translation, UC translates to upper case, and IC inverts the case of alphabetic letters input).

fig-FORTH also defined the word ?TERMINAL to see if a key had been input. This was used in words like VLIST, INDEX, TRIAD and others to see if the user wanted to stop the output on the screen. However, if the user just wanted to stop the output temporarily and then resume it, there was no way to do this.

To get around this problem, I have redefined the word ?TERMINAL to only return a true value if a cntl-C is pressed. If any other key is pressed, ?TERMINAL just waits for another keystroke. This is very handy for looking at the output of a VLIST or for suspending compilation of multiple screens (after I redefined --> to use ?TERMINAL).

<table>
<thead>
<tr>
<th align="left">02 VARIABLE</th>
<th>c</th>
<th>( invert keyboard case as default )</th>
</tr>
</thead>
<tbody>
<tr>
<td align="left">: UC</td>
<td>1 c !</td>
<td>( set case conversion to lower/uppper )</td>
</tr>
<tr>
<td align="left">: LC</td>
<td>0 c !</td>
<td>( set no case conversion )</td>
</tr>
<tr>
<td align="left">: IC</td>
<td>2 c !</td>
<td>( set case conversion to invert case )</td>
</tr>
<tr>
<td align="left">: KEY</td>
<td>( --- key</td>
<td></td>
</tr>
<tr>
<td align="left">(KEY)</td>
<td>( GET NEXT KEYSTROKE</td>
<td></td>
</tr>
<tr>
<td align="left">c @</td>
<td>( ANY TRANSLATION?</td>
<td></td>
</tr>
<tr>
<td align="left">IF DUP 060 &gt; OVER 07B &lt; AND ( CHECK IF LOWER CASE</td>
<td></td>
<td></td>
</tr>
<tr>
<td align="left">c @ 1- IF</td>
<td>( c=2? IS CASE INVERSION</td>
<td></td>
</tr>
<tr>
<td align="left">OVER DUP 040 &gt; SWAP 05B &lt; AND ( CHECK IF UPPER CASE</td>
<td></td>
<td></td>
</tr>
<tr>
<td align="left">OR IF</td>
<td>( UPPER OR LOWER CASE LETTER?</td>
<td></td>
</tr>
<tr>
<td align="left">BL XOR THEN</td>
<td>( YES, INVERT CASE</td>
<td></td>
</tr>
<tr>
<td align="left">ELSE IF</td>
<td>( c=1, LOWER CASE</td>
<td></td>
</tr>
<tr>
<td align="left">BL -</td>
<td>( YES, CONVERT TO UPPER</td>
<td></td>
</tr>
<tr>
<td align="left">THEN THEN THEN ;</td>
<td></td>
<td></td>
</tr>
<tr>
<td align="left">: ?TERMINAL</td>
<td>( --- flag</td>
<td></td>
</tr>
<tr>
<td align="left">FALSE (?KEY)</td>
<td>( KEYSTROKE WAITING?</td>
<td></td>
</tr>
<tr>
<td align="left">IF KEY 3 =</td>
<td>( YES, GET IT, IS IT CNTL-C?</td>
<td></td>
</tr>
<tr>
<td align="left">IF DROP TRUE</td>
<td>( YES, RETURN TRUE</td>
<td></td>
</tr>
<tr>
<td align="left">ELSE KEY 3 =</td>
<td>( NO, WAIT FOR NEXT KEY, =CNTL-C?</td>
<td></td>
</tr>
<tr>
<td align="left">IF DROP TRUE</td>
<td>( YES, RETURN TRUE</td>
<td></td>
</tr>
<tr>
<td align="left">THEN THEN THEN ;</td>
<td></td>
<td></td>
</tr>
<tr>
<td align="left">: --&gt;</td>
<td>( ---</td>
<td></td>
</tr>
<tr>
<td align="left">?LOADING ?TERMINAL</td>
<td>( CHECK FOR CNTL-C</td>
<td></td>
</tr>
<tr>
<td align="left">IF QUIT THEN</td>
<td>( YES, QUIT LOADING</td>
<td></td>
</tr>
<tr>
<td align="left">O &gt;IN !</td>
<td>( RESET &gt;IN</td>
<td></td>
</tr>
<tr>
<td align="left">B/SCR BLK @ OVER MOD</td>
<td>( # BUFFERS IN BLOCK PARSED SO FAR</td>
<td></td>
</tr>
<tr>
<td align="left">- BLK +! ; IMMEDIATE</td>
<td>( INC BLK BY AMOUNT LEFT</td>
<td></td>
</tr>
</tbody>
</table>
Select, Ordered, Perform

Wil Baden
Costa Mesa, California

Another application of Quicksort is based on the “Simple Files” example in Leo Brodie’s Starting Forth. This allows selection and display of sorted records.

Three end-user words are provided — SELECT, ORDERED and PERFORM. Here are examples of usage:

SELECT JOB NEWSCASTER

will select all records where the JOB field is NEWSCASTER.

ORDERED SURNAME

will sort them by surname (after a SELECT).

PERFORM GET GIVEN
GET SURNAME GET PHONE

will print an alphabetic phone list (after a GET SURNAME GET PHONE PERFORM GET GIVEN).

Another application of Quicksort is beginning with #RECORD and proceeding down, compares the contents of the field indicated by KIND against the contents of WHAT.

TEXT (addr1, len, addr2 -- n)

compares strings at addr1 and addr2 for length of len, and returns negative, zero or positive value for the string at addr1 (for less-than, equal-to or greater-than the string at addr2).

FIELD (addr1 -- addr2, len)
given the address of a field-specifying table, ensures that the associated field in the current record is in a disk buffer, and returns the address of the field in the buffer along with its length.

GET ( -- <fieldname>)

prints the contents of the given type of field from the current record.

KEEP ( addr -- <string>)

moves a character string, delimited either by a comma or by a carriage return, from the input stream into WHAT, and saves the address of the given field-specifying table into KIND for future use by -FIND.

KIND ( -- addr)
is a variable that contains the address of the field-specifying table for the type of field that was last searched for by -FIND.

MAXRECS ( -- n)
is the maximum number of records to be allowed in the system.

RECORD ( -- addr)

ensures that the current record is in a disk buffer, and returns the address of the first byte of that record.

TOP ( -- )

resets the record pointer to the top of the file.

WHAT ( -- addr)

returns the address of a buffer that contains the string that is being searched for, or was last searched for.

Note: for the full glossary and the source for the simple file system, see Leo Brodie’s Starting Forth, pp. 328 – 340.
Interrupt service routines are brief (fast executing) routines which are performed in 1/60 second or less. While operating in Forth, user-defined ISRs are called after each keyboard scan without regard to whether a key was pressed, and after each call to the NEXT instruction. (All CODE words, such as MON, DOOT and SMASH terminate with NEXT.) Because the ISR executes in sixteen milliseconds or less and is called so often, it appears to the operator as instantaneous with the current foreground task is called so often, it appears to the operator as instantaneous with the current foreground task and instantaneous in reaction time.

The ISR Base-display (screen 41) displays the current BASE in decimal while in command mode. When BASE is changed, the effect is instantaneous, yet no keyboard scans or other computing tasks that your machine may be performing will be missed. Before installing the ISR, there are several changes that I recommend you make. These changes will set up your system to execute the ISR only when not in TEXT, GRAPHICS, MULTI-MODE, SPLIT2 or SPLIT (when not being used for editing). In other words, the ISR will only be executed while you are editing. The following screen numbers refer to your copy of the TI-FORTH master disk.

Screen 22, line 1
: EDT ISR VDPMDE @ . . .

Screen 22, line 11
... CLS SCRNO ISR DROP . . .

Screen 33, line 15
DECIMAL "XX" CLOAD ISR R->BASE

Where "XX" is the screen on which you put the ISR Base-display.

Screen 38, line 1
: VED ISR BOX SWAP . . .

Screen 38, line 2
OF OF ISR BCK . . .

Screen 51, line 4
: TEXT ISR . . .

Screen 52, line 4
: GRAPHICS ISR . . .

System is in machine language. 2 SYSTEM will expect on the stack the location/destination addresses and byte count of the write. Further, for our needs, the numbers to be written must be converted to ASCII before being pushed into the VDP buffer BASE. That conversion is accomplished on line 5. For example,

16 10 /MOD . . . 16 ok

Since a division remainder can never be larger than the divisor, the effect of 10 /MOD is to leave on the stack the dividend split into its component divisors. Adding 48 to each of them produces the ASCII codes needed. They are then pushed into the two bytes of the buffer BUF, where 2 SYSTEM will get them.

Lines 7~10 write the desired information directly into the VDP screen table. All of this, of course, will occur in less than sixteen milliseconds.

Address hex 83C4 is the user-defined ISR pointer, where zero is off. ISR is a user

SCR #41
0 ( ISR Base-display installation, Gene Thomas, Jul85) BASE->R
1 DECIMAL 39 CLOAD SCOPY (necessary for the !" definition)
2 0 VARIABLE BAS (radix buffer for vdp write)
3 0 VARIABLE STR (string buffer for vdp write)
4 STR (push string into buf)
5 : BAS (-> BASE @ 10 /MOD 48 + SWAP 48 + BAS 1+ C! BAS C!
6 (convert base to ASCII and store in vdp radix buf)
7 STR SCRN_START @ 25 + (read from/write to addr's (string))
8 5 2 SYSTEM write 5 bytes (Base?)
9 BAS SCRN_START @ 20 + (read from/write to addr's (radix))
10 2 2 SYSTEM write 2 bytes (##) ;
11 HEX
12 0 ISR (-- ip ISR off) 0 B3C4 !
13 : ISR (-- ip ISR on) INTLNK @ B3C4 !
14: INSTALL-ISR (-- ip followed by WORD to be installed)
15 0 ISR [COMPILE] ' BL WORD CFA ISR ; R->BASE

Volume VIII, No. 1 23 FORTH Dimensions
variable which contains the CFA of the ISR to be executed. (The TI-99/4A executes on CFA's, and not on PFAs as do some systems.) INSTALL-ISR is another user variable. Placing the contents of INTLNK into 83C4 turns the ISR on. When the CFA of the routine to be executed as an ISR is pushed into the variable ISR, INTLNK is automatically updated. It is important that CFA ISR ! be accomplished before the ISR is set to on by pushing the contents of INTLNK into 83C4! Thus, INSTALL-ISR begins by turning the ISR off (0ISR).

Here is the sequence of commands that will get the ISR going:
INSTALL-ISR 1ISR

There are two ways to turn it off. 0ISR will do it; however, if you have made the recommended changes to the copy of the master disk, it will be turned on again each time you leave edit with the FCTN BACK keys. A better way to turn it off is to install INSTALL-ISR OISR (OISR).

That is a fair question. The straightforward answer is, you don't. But there are some guidelines. With the ISR installed, there should be no problem with normal keyboard or program execution; if there isn't, it should be running fast enough. Avoid deeply nested routines, such as long IF THEN and CASE constructs, and slow-running operations such as the floating-point routines. Any output conversion process (D., ., EMIT, TYPE, SPACES, etc.) will certainly be too slow. You should be successful if you learn to use the following words to their best advantage: VMBW, YSBW, YSBR, YMBR, AND, OR, XOR, MOD, UI, +!, SLA and SRA. These words and some others call directly, or nearly so, on machine code.

Finally, you should be using -64SUPPORT for editing if you have a monitor or a TV with good resolution. If you are using the forty-column screen editor, then you'll want to change the 25 + and 30 + following SCRN-START at in BAS? I suggest 31 and 38 respectively, to put the display in the upper right corner.

THE TOOLS GROUP
66230 Forth Street
Desert Hot Springs, CA 92240
619/329-4625

Do you use Forth professionally?
Do you ever wish that you were working in an organization large enough to have a full time software tools group?

The Tools Group subscription support service is available to any Forth programmer. Tools are available for the Z80 and MC680x0 environments, running either CP/M 2.2, TPM 111, or GEMDOS. iAPX 8x/28x and NS32x32 versions will be ready soon.

Access to the Tools Group programmers and library is through a Hartronix multiuser bulletin board system. Soon, the service will be available worldwide through an X.25 based public data network at very low cost.

The Tools Group library is extensive, including tools like a 64 bit IEEE floating point package, transportable code between different operating systems, ASCII file support, automatic library manager, and much more. If the tool you need is not in the library, we will work with you to develop it.

Our motives are simple. We enjoy building real tools for real programmers. We want to help you. For a small annual fee, you may freely use our tools without royalties.

You don't need to tell anyone that you have a tools group -- you can just let them think you are Superprogrammer.
Moore Chats on CompuServe

Editor's note: FIG member Ward McFarland recently mailed us the transcript of a public meeting held one evening on CompuServe. Making a prearranged guest appearance, Mr. Charles Moore fielded questions from others who were simultaneously logged on to the electronic information network. The inventor of the Forth language had a good deal to say about the Novix Forth chip he designed — a topic of interest to many of our readers — and also touched on several other topics. See our "Letters" section for more about Forth on CompuServe, and watch upcoming issues for a roundup of all electronic sources of Forth-related information. The following transcript has been edited for readability, but we have left the cryptic "citizen's band" style mostly intact, as it is representative of what attendees at such live, electronic meetings can expect to find.

(Don Colburn) Good evening, and welcome to the CSI Forth Net. I'm Don Colburn, and I'll be your moderator. We come to you tonight live from the hills above Silicon Valley, south of San Francisco, California.

Tonight I am very pleased to welcome Charles Moore, the inventor of Forth, as our guest. Topics for this evening include general questions on Chuck's unique perspective on the relationship between men and computers, as well as specific questions about the newly designed Novix Forth microprocessor that he has designed for Novix.

Before we get started, I'd like to review how this conference will be conducted. As Moderator, I'll begin by asking for questions from the audience. If you have a question, type a question mark followed by a carriage return. When you enter the question mark on your terminal, I'll see it here on my screen, along with your name. I'll note your name from the question marks that were received here. By jotting down your question before asking it, you can help the conference run a lot more smoothly. If this is your first conference on CompuServe, relax and watch the traffic for a few minutes. After you've seen a couple of questions and answers go by, you'll get the hang of it.

Let's get started. Does anyone have a question for Chuck?

(Scott S.) Any comments on the Delta board from Software Composers (I understand you won't be making a Gamma board) and any comments on CMForth that is on the Delta board? (CMForth is Charles Moore's Forth.)

(Chuck Moore) Delta board is a bargain. Made in Taiwan for a price no one can beat. Ten times the speed of any PC.

(Scott S.) Heard that the CMForth was very interesting and that you had simplified the Novix compiler. It was recommended to me for everyone, whether they bought a Novix or not (last Silicon Valley FIG Chapter meeting).

(Chuck) CMForth is the absolute simplest system I can conceive. On the other hand, I have a much simpler system in the wings. It does well illustrate the Novix chip.

(Dave S.) For those of us who are ignorant, what's a Delta board?

(Chuck) A 4" x 6" board with the Novix chip and 10K words of memory. Fully stuffed and tested (?), it sells for about $900. It talks to a host computer over a serial line and runs Forth fast.

(Dave S.) Sounds like it's meant as a controller board. Is that a reasonable assumption?

(Chuck) The key feature is that it can recompile itself. Thus, it is a complete development system for Novix applications.

(Scott S.) I can upload a text file with the specs they hand out. I think they had it for the 138, it won't be needed soon. Novix board for expansion. 20 MHz video output takes four chips, and input requires two.

(Dave S.) I think we'd like to have that, Scott. Actually, that was not the original question I had in mind (obviously). But maybe we should give others a chance.

(Chuck) A little more on the Delta board. Parts count is minimal: eight memory chips, 4 MHz oscillator and a 74138. I apologize for the 138, it won't be needed soon. Novix chip accesses sixty-four bits of memory every cycle (at best). Selected chips can run at 10 MHz (?). Four busses are taken off board for expansion. 20 MHz video output takes four chips, and input requires two more.

(John B.) Chuck, as a reasonably long-time Forth user (1979), let me say, "Thanks for the language!!" By the way, if Ada ever truly is "The Standard Language," I'm going sailing!

(Don Colburn) Save me a berth!
I'm facing a problem of implementing a three-stack threaded language of my own on fast hardware, either Novix or Metaforth NF16LP. The language is for discrete event simulation and floating-point number crunching. I have difficulty choosing between Novix and Metaforth; both are fast enough. What would you consider to be a decisive factor?

(Chuck) Novix exists! A third stack is easy to emulate with Novix. Floating point should involve a coprocessor (in both cases).

(Kiyoshi Y.) Metaforth has floating. Maybe not HW.

(Chuck) They have board to date. Chip implementation of floating point is not cost effective. But being Forth, they are interesting alternative. What is interest in Forth in Japan?

(Kiyoshi Y.) Not very popular. But I've seen a Forth coprocessor to work with 68000 made in Kyoto. Several books exist to teach how to program in Forth.

(Chuck) Interesting.

(Scott S.) When will Novix make the next batch of the current chip, when will the "next" generation chip be, how are the interrupt problems coming?

(Chuck) I heard today that fifty-nine are in test. 441 will follow shortly. How can we sell them?

(Scott S.) I think the Delta board is a good first step. Would like to see an I/O-type system with interrupts.

(Chuck) Interrupts are usable, if you expect them, i.e., as synchronizing signals. Next chip is under consideration. Many changes are possible, but desirable? Say six months. Buy current chip!

(Scott S.) OK, I will! Just one more quick question. Heard that a company was connecting the Weitek math chips, and can you tell us what some companies are planning to do with the chips themselves.

(Chuck) Attractive for telecommunications, digital audio, process control. Attaching math chips, or any chips, is very easy. So easy that on-chip capability is questionable. But no names at present.

UBZ FORTH™
for the Amiga™

*FORTH-83 compatible
*32 bit stack
*Multi-tasking
*Separate headers
*Full screen editor
*Assembler
*Amiga DOS support
*Intuition support
*ROM kernel support
*Graphics and sound support
*Complete documentation
*Assembler source code included
*Monthly newsletter

$85
Shipping included in continental U.S. (Ga. residents add sales tax)
UBZ Software
(404)-948-4654
(call anytime)
or send check or money order to:
UBZ Software
395 St. Albans Court
Mableton, Ga. 30059

*Amiga is a trademark for Commodore Computer. UBZ FORTH is a trademark for UBZ Software.
(Jim M.) Concerning popularity, why not invite academia a la Pascal?

(Chuck) Would love to. But, historically, universities haven’t had the right hardware. Now we can provide (give) computers and the situation might change. More academics are discovering Forth.

(Jim M.) Good. We need to convince them of Forth’s special capabilities.

(Kiyoshi Y.) What is the reason for not making Forth syntax completely postfix?

(Chuck) Clarify. Forth is postfix, except for: which is necessary.

(Kiyoshi Y.) What about WORD, FORGET, ...

(Chuck) I think strings must have prefix to avoid confusion. But I welcome suggestions.

(Kiyoshi Y.) STOIC employs a single quote before strings.

(Chuck) This leads to lots of useless quotes.

(Kiyoshi Y.) But the analyzer becomes simpler, it seems.

(Chuck) No. The simple convention that strings are prefix saves lots of effort. Actually, I’m not a purist. Use what works. I do equate simplicity with minimum character count.

(Kiyoshi Yoneda) I see. But uniformity leads to less confusion on user’s part. Thank you.

(Chuck) Confusion exists once, upon learning. Cost exists forever.

(Ward) That would be welcomed by many, I think. On another topic, what is your opinion of highly machine-dependent Forths (like MacForth), which add many useful extensions at the expense (?) of portability?

(Chuck) I don’t value portability. I’ve seen very few examples of applications being ported when they couldn’t have been edited. Give me the central idea of the application and I’ll embed it in my own code.

(Kiyoshi Y.) Contact address or telex for Delta?

(Don Colburn) Software Composers (Delta Card) phone number 415-327-6891.

(Michael B.) Speaking of code, are you still writing one- to two-line words?

(Chuck) More than ever. The factoring of an application leads to the development of a language. The language is more important than the application, and rarely receives the appreciation it deserves. A concept that cannot be expressed tersely is a badly understood concept.

(Michael B.) Words to the wise (who will listen) — thank you.

(Dave S.) As an academic, I’m interested in FORTH becoming more popular at universities. But I wonder, why do you think free hardware would help? We get more free stuff than we know what to do with, from tax write-offs.

(Chuck) Free is not sufficient, but necessary. Universities, in my experience, have no money to spend for hardware or software. But they have graduate students. To expect a graduate student to construct a FORTH system is asking a lot. To ask him to assemble a kit, one or two days, may be reasonable if he gets something he couldn’t buy. I wish he would be motivated to try FORTH without encouragement, but he hasn’t been.

NEW PRODUCTS

FOOTSTEPS IN AN EMPTY VALLEY -- NC4000 Single Chip Forth Engine 2nd Edition, Apr 1986 C. H. Ting First comprehensive book on the Novix NC4000 chip. Architecture, instruction set, circuit diagram of a single board computer, description of cmFORTH operating system, programming tips, source code of cmFORTH, glossary, and index. $25.00

NC4000 Microprocessor Super fast Forth engine from Novix, Inc. $250.00 ppd Quantity discount available.

BUILD YOUR OWN ELECTRONIC ORGAN C. H. Ting Use an IBM-PC to build an organ! Instructions and source code to build and play an organ with up to 12 voice channels. Great for Bach’s polyphonic organ music. $15.00

PARALLEL INTERFACE CARD FOR PC Four 8253’s and four 8255’s with 12 counter-timers and 96 digital I/O lines. Great for above organ and robotic applications. $159.00

OFFSET ORGAN SERIES C. H. Ting Source and COM files on IBM-PC disks. J. S. Bach’s music. Disk I. Inventions, preludes, fugues and Toccata In G Major. Disk II. 24 pieces in Anna Magdalena’s Notebook. Disk III. Selected cantata chorales for organ. $15.00 per disk

OTHER TITLES

INSIDE F83 C. H. Ting Full documentation of F83 system $25.00

FORTH NOTEBOOK C. H. Ting Games and applications. $25.00

SYSTEMS GUIDE TO fig-FORTH Ting Full documentation of fig-FORTH. $25.00

F83 DISK LIBRARY John Peters Collection of source code. $20.00

F83 SOURCE H. Laxen & M. Perry For 8080, 8086 and 68K versions. $25.00

FigAI NOTES C. H. Ting AI working group report. $15.00

F83 DISKS H. Laxen & M. Perry For PC, CP/M, CP/M86, CPM/68K. Please specify your OS and disk format. $25.00 per disk.

F83X DISK Wil Baden For Apple II computer. $25.00

Send check or money order to: Offsets Enterprises, Inc. 1306 S. B St., San Mateo, Ca. 94402 Mailing & Handling, 10% of order. Californians please add 6.5% sales tax.
FOR TRS-80 MODELS 1, 3, 4, 4P IBM PC/XT, AT&T 6300, ETC.

Train Your Computer to be an EXPERT!

Expert systems facilitate the reduction of human expertise to simple, English-style rule-sets, then use them to diagnose problems. "Knowledge engineers" are developing many applications now.

EXPERT-2, Jack Park's outstanding introduction to expert systems, has been modified by MMS for MMSFORTH V2.0 and up. We supply it with full and well-documented source code to permit addition of advanced features, a good manual and sample rule-sets: stock market analysis, a digital fault analyzer, and the Animal Game. Plus the benefits of MMSFORTH's excellent full-screen editor, super-fast compiling, compact and high-speed run-time code, many built-in utilities and wide choice of other application programs.

(Dave S.) I think graduate students do try Forth — and everything else, too. But that won't make it popular. The language taught to undergrads, where the big numbers are, is Pascal because of its "save the programmer from himself" philosophy, which makes teaching easier. Comments?

(Chuck) Wow! Pascal, in the world I encounter, is a joke. Fortran is the language of choice. Not choice, necessity. This shocks me to the bone. I thought Fortran was passe. The programmer needs protection or, rather, the system needs protection on a multi-programmed computer, but that situation no longer obtains. The joy of crashing the system must be experienced to be appreciated. Forth should be taught to junior high school students before they learn algebra.

(Dave S.) Yes, engineering students still get Forth instead of Pascal. But here, all CompSci students get Pascal. Used to be PL/C until recently.

(Scott S.) A group is putting Pascal on the Novix. I've heard that there is also a group putting C on the Novix. Do you know about this?

(Chuck) I know of C, Novix is involved. Other languages (Lisp, COBOL, BASIC) are being addressed. I question the value. The latest insight into the chip's architecture leads me to think that Forth is quintessentially right. The other languages can't come within an order of magnitude of its performance — for intrinsic reasons.

(Scott S.) Would be worth it just to see their expression — ha! Especially since some Forths on Unix machines are written in C. Any thoughts on artificial intelligence?

(Chuck) AI is well suited to Forth. Not popularly. I'd like to see AI solve a problem (locomotives not counted). I think I can recognize connected speech with the chip, without AI.

(John B.) Before I found Forth, I was already using HP-25 etc., so I was prepared for stack/postfix. Wonder how HP vs. TI experience correlates with acceptance vs. rejection of Forth upon trial. (I still can't use a TI calculator effectively!)

(Chuck) We have interest from both TI and HP. My son can't use TI either.

(David B.) One problem is people are getting unsupported Forths for free, then being disappointed by the lack of support! (Or just bewildered, without a manual, by how strange Forth looks.) This initial "bad" encounter with Forth may have done much to hamper Forth's growth.

(Chuck) That's the problem with grassroots growth. If I sell a Gamma board, it will be without support. Not because I don't appreciate the need for support, but because I can't provide it. Is an unsupported product better than nothing? Even if I direct it at a knowledgeable audience? Even if no choice? You may be right, but what are the alternatives?

(David Butler) That's the issue. Forth seeks a knowledgeable audience, but some need initial hand holding before they see the light.

(Kiyoshi Y.) Japan is potentially a big Forth market, because our grammar is exactly postfix.

(Chuck) Funny how great minds think alike. Your comment concisely summarized our local discussion here. I see Japan as a great competitor. I don't wish to aid her. But she would be well advised to pay attention to Forth. Prolog will fail. If the fifth generation depends on Prolog, it will fail. Is that good?

(Kiyoshi Y.) I agree. Prolog will fail. The "Fifth" is falling apart. As for the aid, I think you've already helped us a lot.

(Don Colburn) Perhaps we would trade lawyers for Forth, but you would have to take both.

(Michael B.) Do you think the fifth generation could be done using Forth?
(Chuck) Oy! I think Forth is the only possibility. Even stronger, Forth is the only SDI possibility. The chance of convincing them of that is zero. So pray for disarmament.

(Michael B.) How many Delta boards did you say you had — build a prototype!

(Chuck) Delta boards are similar to Gamma boards. Gamma boards nest. That is, they plug into each other. Sure, could construct anything, but who would notice?

(Michael B.) A working prototype could be sold (or traded for lawyers) to the Japanese.

(Chuck) Good point, and an interesting closing thought. Oops, go ahead JM, you are the last.

(Jim M.) I suggest that the best way to convince the newcomer of Forth's power is to hold some kind of contest. Lock several programmers in a closet with a problem and see who solves it first. I think the Forth programmer would win by a landslide. Comments?

(Don Colburn) An even more provocative thought. Sounds like a good thread in the special topics section of the Net. Let's hear suggestions of an appropriate problem and an appropriate reward.

OK, time to go. Thanks, Chuck, and thanks to everyone for attending this evening. If you would like to do this again sometime, or have an idea for another conference, please leave a message to any of the sysops. We would love to hear from you.

Good night.
U.S.

- **ALABAMA**
  Huntsville FIG Chapter
  Call Tom Konantz
  205/881-6483

- **ALASKA**
  Kodiak Area Chapter
  Call Horace Simmons
  907/486-5049

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

- **ARKANSAS**
  Central Arkansas Chapter
  Twice Monthly, 2nd & 4th Sun., 2 p.m.
  Flexible Hybrid Systems
  2030 E. Broadway
  Call John C. Mead
  602/323-9763

- **CALIFORNIA**
  Bay Area Chapter
  Silicon Valley Chapter
  Monthly, 4th Sat.
  FORML 10 a.m., Fig 1 p.m.
  H-P Auditorium
  Wolfe Rd. & Pruneridge,
  Cupertino
  Call John Hall 415/532-1115
  or call the FIG Hotline:
  408/277-0668

  Stockton Chapter
  Call Doug Dillon
  209/931-2448

  Stockton Chapter
  Call Doug Dillon
  209/931-2448

  - **COLORADO**
    Denver Chapter
    Monthly, 1st Mon., 7 p.m.
    Cliff King
    303/693-3413

  - **CONNECTICUT**
    Central Connecticut Chapter
    Call Charles Krajewski
    203/344-9996

  - **FLORIDA**
    Orlando Chapter
    Every two weeks, Wed., 8 p.m.
    Call Herman B. Gibson
    305/855-4790

    Southeast Florida Chapter
    Monthly, Thurs., p.m.
    Coconut Grove area
    Call John Forsberg
    305/252-0108

  - **GEORGIA**
    Atlanta Chapter
    3rd Tuesday each month, 6:30 p.m.
    Computone Cottilion Road
    Call Ron Skelton
    404/393-8764

  - **ILLINOIS**
    Cache Forth Chapter
    Call Clyde W. Phillips, Jr.
    Oak Park
    312/386-3147

    Central Illinois Chapter
    Urbana
    Call Sidney Bowhill
    217/333-4150

    Fox Valley Chapter
    Call Samuel J. Cook
    312/679-3242

    Rockwell Chicago Chapter
    Call Gerard Kusiolek
    312/885-8092

  - **INDIANA**
    Central Indiana Chapter
    Monthly, 3rd Sat., 10 a.m.
    Call John Oglesby
    317/353-3929

  - **IOWA**
    Iowa City Chapter
    Monthly, 4th Tues.
    Engineering Bldg., Rm. 2128
    University of Iowa
    Call Robert Benedict
    319/337-7853

    Central Iowa FIG Chapter
    Call Rodrick A. Eldridge
    515/294-5659

  - **KANSAS**
    Wichita (FIGPAC)
    Monthly, 3rd Wed., 7 p.m.
    Call Gurdv Leete
    316/472-7077

  - **LOUISIANA**
    New Orleans Chapter
    Call Darryl C. Olivier
    504/899-8922

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

  - **MICHIGAN**
    Detroit Chapter
    Monthly, 4th Wed.
    Call Tom Chrapkiewicz
    216/336-3167

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

  - **MISSOURI**
    Kansas City Chapter
    Monthly, 4th Tues., 7 p.m.
    MidWest Research Institute
    MAG Conference Center
    Call Linus Orth
    913/236-9189

  - **MISSOURI**
    St. Louis Chapter
    Monthly, 1st Tues., 7 p.m.
    Thornhill Branch Library
    Contact Robert Washam
    91 W. Weis Dr.
    Ellisville, MO 63011

  - **NEVADA**
    Southern Nevada Chapter
    Call Gerald Hasty
    702-452-3308

  - **NEW HAMPSHIRE**
    New Hampshire Chapter
    Monthly, 1st Mon., 6 p.m.
    Armtec Industries
    Shephard Dr., Grenier Field
    Manchester
    Call M. Peschke
    603/774-7762

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

  - **NEW YORK**
    FIG, New York
    Monthly, 2nd Wed., 8 p.m.
    Queens College
    Call Ron Martinez
    212/517-9429

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

    Rockland County Chapter
    Call Elizabeth Gormley
    Pearl River
    914/735-8967

    Syracusse Chapter
    Monthly, 3rd Wed., 7 p.m.
    Call Henry J. Fay
    315/446-4600

  - **OHIO**
    Akron Chapter
    Call Thomas Franks
    216/336-3167

    Athens Chapter
    Call Isreal Urieli
    614/594-3731

    Cleveland Chapter
    Call Gary Bergstrom
    216/247-2492

    Cincinnati Chapter
    Call Douglas Bennett
    513/831-0142

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

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

Oregon
Greater Oregon Chapter
Monthly, 2nd Sat., 7 p.m.
Tektronix Industrial Park
Call Tom Admy
503/692-2811

Philadelphia Chapter
Monthly, 4th Sat., 10 a.m.
Drexel University, Stratton Hall
Call Melanie Hoag or Simon Edkins
215/895-2628

Pennsylvania
Philadelphia Chapter
Monthly, 4th Sat., 10 a.m.
Drexel University, Stratton Hall
Call Melanie Hoag or Simon Edkins
215/895-2628

Tennessee
East Tennessee Chapter
Monthly, 2nd Tues., 7 p.m.
800 Oak Ridge Turnpike, Oak Ridge
Call Richard Scrisiti
615/483-7242

Texas
Austin Chapter
Contact Matt Lawrence
P.O. Box 180409
Austin, TX 78718

Dallas/Ft. Worth
Metroplex Chapter
Monthly, 4th Thurs., 7 p.m.
Call Chuck Durrett
214/245-1064

Houston Chapter
Call Dr. Joseph Baldwin
713/749-2120

Periman Basin Chapter
Call Carl Bryson
Odessa
915/337-8994

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

Vermont
Vermont Chapter
Monthly, 3rd Mon., 7:30 p.m.
Vergennes Union High School
Rm. 210, Monkton Rd.
Vergennes, VT
Call Don VanSycel
802/388-6698

Virginia
First Forth of Hampton Roads
Call William Edmonds
804/898-4099

Potomac Chapter
Monthly, 2nd Tues., 7 p.m.
Lee Center
Lee Highway at Lexington St.
Arlington, VA
Call Joel Shprenitz
703/860-9260

Richmond Forth Group
Monthly, 2nd Wed., 7 p.m.
154 Business School
Univ. of Richmond
Call Donald A. Full
804/739-3623

Wisconsin
Lake Superior FIG Chapter
Monthly, 2nd Fri., 7:30 p.m.
University of Wisconsin Superior
Call Allen Anway
715/394-8360

Milwaukee Area Chapter
Call Donald H. Kimes
414/377-0708

MAD Apple Chapter
Contact Bill Horzon
129 S. Yellowstone
Madison, WI 53705

foreign
Australia
Melbourne Chapter
Monthly, 1st Fri., 7 p.m.
Contact Lance Collins
65 Martin Road
Glen Iris, Victoria 3146
03/29-2600

Sydney Chapter
Monthly, 2nd Fri., 7 p.m.
John Goodsell Bldg.
Rm. LG19
Univ. of New South Wales
Sydney
Contact Peter Tregeagle
10 Binda Rd., Yowie Bay
02/524-7490

Belgium
Belgium Chapter
Monthly, 4th Wed., 20:00h
Contact Luk Van Loock
Lariksdruff 10
2120 Schoten
03/658-6343

Southern Belgium FIG Chapter
Contact Jean-Marc Bertincamps
Rue N. Monnom, 2
B-6290 Nalinnes
Belgium
071/213858

Canada
Alberta Chapter
Call Tony Van Myden
403/962-2203

Nova Scotia Chapter
Contact Howard Harawitz
227 Ridge Valley Rd.
Halifax, Nova Scotia B3P2E5
902/477-3665

Southern Ontario Chapter
Quarterly, 1st Sat., 2 p.m.
General Sciences Bldg., Rm. 312
McMaster University
Contact Dr. N. Soltesz
Unit for Computer Science
McMaster University
Hamilton, Ontario L8S4K1
416/525-9140 ext. 3443

Toronto FIG Chapter
Contact John Clark Smith
P.O. Box 230, Station H
Toronto, ON M4C5J2

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

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

France
French Language Chapter
Contact Jean-Daniel Parra B.
Aptdo. Aereo 100394
Bogota
214-0345

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

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

Ireland
Irish Chapter
Contact Hugh Doggs
Newton School
Waterford
051/77573 or 051/74124

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

Japan
Japan Chapter
Contact Toshi Inoue
Dept. of Mineral Dev. Eng.
University of Tokyo
7-3-1 Hongo, Bunkyo 113
812-211 ext. 7073

Norway
Bergen Chapter
Kjell Birger Faeruaas
Hallsavarei 28
Ulstein + 47-5-187784

Republic of China
R.O.C.
Contact Ching-Tang Tseng
P.O. Box 28
Lung-Tan, Taiwan 325

Sweden
Swedish Chapter
Hans Lindstrom
Gothenburg
+46-31-166794

Switzerland
Swiss Chapter
Contact Max Hugelshofer
EERNI & Co., Elektro-Industrie
Stationstrasse
8306 Bruttisellen
01/833-3333

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

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

FIGGRAPH
Call Howard Pearlmutter
408/425-8000

Forth Dimensions
Proceedings of the 1985 FORML Conferences

containing papers from the

Seventh Asilomar FORML Conference
and the
euroFORML Conference

Forty-five Forth articles in one volume with 480 pages.

Partial List of Topics

Applications (Expert Systems, data collection, networks)
Languages (Lisp, LOGO, Prolog, BNF)
Style (Coding conventions, phrasing)
Software Tools (decompilers, structure charts)
Forth Internals (Forth computers, floating point, interrupts, multitasking, error handling)

Recommended for professional engineers and programmers using Forth.

Order now from the Forth Interest Group - $35. Use order form enclosed.

FORTH INTEREST GROUP
P. O. Box 8231
San Jose, CA 95155