FORTH Dimensions

Volume 8, Number 1

May/June 1986
$4.00

Serial I/O & Interrupts

Fast Fixed-Point Trig

Case Conversion in KEY

Select, Ordered, Perform

Moore Chats on CompuServe

Forth Dimensions
Published by the
Forth Interest Group
Volume VIII, Number 1
May/June 1986
Editor
Marlin Ouverson
Production
Cynthia Lawson Berglund
Typesetting
LARC Computing

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

Subscription to *Forth Dimensions* is free with membership in the Forth Interest Group at $30 per year ($43 foreign air). For membership, change of address and to submit material for publication, the address is: Forth Interest Group, P.O. Box 8231, San Jose, California 95155.

Symbol Table

Simple; introductory tutorials and simple applications of Forth.

Intermediate; articles and code for more complex applications, and tutorials on generally difficult topics.

Advanced; requiring study and a thorough understanding of Forth.

Code and examples conform to Forth-83 standard.

Code and examples conform to Forth-79 standard.

Code and examples conform to fig-FORTH.

Deals with new proposals and modifications to standard Forth systems.

FORTH Dimensions

FEATURES

DEPARTMENTS

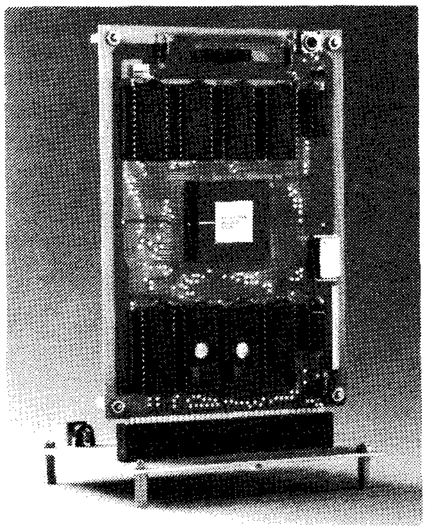
THEY'RE HERE!

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 I never considered. The whole thing is well done.

After reading Michael Hore's excellent letter (VII/3) about standards for Forth on improved micros, I have some alternate 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** . . . **I** @ . . . **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** . . . **I** @ . . . **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 randon 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. ! 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 Yngve's "Compiler Macros" as a follow-up to his "Synonyms and Macros" series.

FIG SYNONYM

Listing 1

```
Screen #5
 \ Synonyms: FD vol 3, no 3;                    Forth-83 11/28/84 vhy
 \ Converted to FIG                                       gt Apr 86
 : SYNONYM   ( new old -- )
    <BUILDS
    -FIND                               \ found: pfa cnt tf; ff
    IF DROP ( count)  DUP CFA ,
      IMMEDIATE                         \ make new immediate
      DUP CFA SWAP  NFA  64 AND         \ was old immediate?
         IF  DOES>  @ EXECUTE           \ yes, set new to execute
           ELSE  DOES> STATE @          \ no, set to check state
           IF @ ,                       \ compile if compiling
             ELSE @ EXECUTE             \ or execute if executing
           THEN
         THEN
      ELSE CR ." Not found"  ABORT      \ old not found
    THEN  ;
```

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)

Editorial

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, 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 MacForthers. 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.

Len Zettel
Trenton, Michigan

```
SCREEN #64
   LOOKUP
 : LOOKUP ( N1 ADDR1 N2 --- ADDR2) £ ADDR2 = ADDRESS OF VALUE N1
   IN THE TABLE OF N2 ENTRIES STARTING AT ADDR1.
   0 ROT ROT OVER + SWAP
   DO OVER I @ =
      IF DROP I LEAVE THEN
   LOOP
   SWAP DROP ;
```

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

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 **DEBUG** command provided with the operating system. For example, if you are running F83, use **DEBUG 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. (**PC!**, 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-IOFF 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 it, instead of using Forth's **KEY** — especially since some remote systems require a 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 either way. (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
( Interrupt service routine                              4-10-86 )
HEX
VARIABLE PTR   ( Where interrupt routine will put characters )
3FD CONSTANT LSTATUS   ( Line status register )
3F8 CONSTANT RBUFFER  ( Receive buffer )
ASSEMBLER
CREATE IIN  ( Interrupt service routine, Port 1, Input )
   AX PUSH   BX PUSH   DX PUSH   ( Save registers )
   LSTATUS # DX MOV   DX AL IN   01 # AL TEST   ( Data ready? )
   0<> IF   RBUFFER # DX MOV   DX AL IN   ( Read character )
     CS: PTR #) BX MOV   ( Move it to (PTR )
     CS: AL 0 [BX] MOV   BX INC   ( Inc PTR )
     CS: BX PTR #) MOV   THEN
   DX POP   BX POP   ( Restore these registers )
   20 # AL MOV   20 # AL OUT   ( 8259A Interrupt controller )
   AX POP ( Restore )  IRET     FORTH     DECIMAL
```

```
Scr # 3          A:FTERM.BLK
( Interrupt setup                                          4-10-86 )
HEX
3F9 CONSTANT IENABLE   ( Interrupt enable register )
3FC CONSTANT MCONTROL   ( Modem control register )
CODE DOS2    ( dx cx bx ax -- ax2   Negative return if err )
   AX POP  BX POP  CX POP  DX POP  21 INT
   U< IF  ( Carry flag set, meaning error )  AX NEG  THEN
   1PUSH  END-CODE
: ISETUP    ( --    Prepare to receive serial input interrupts )
   IIN 0 0 250C DOS2 DROP    ( Interrupt vector for COM1 )
   1 IENABLE PC!    ( Enable data-available interrupt )
   8 MCONTROL PC! ( Needed ) ;
( Note: finished product should save and restore int. vector )
DECIMAL


Scr # 4          A:FTERM.BLK
( Interrupt on/off                                         4-10-86 )
HEX
( Note - MUST set PTR before turning interrupt on )
( Ports 20 and 21 are control registers for the 8259A )
: ION    ( --    Turn COM1 interrupt on )
   21 PC@  EF AND  21 PC! ;
: IOFF ( --    Turn interrupts off )
   21 PC@  10 OR  21 PC!  ( Set the mask ) ;
DECIMAL


Scr # 5          A:FTERM.BLK
( Read a character - by taking it from buffer          4-10-86 )
VARIABLE LPTR    ( Last value of pointer )
: FILTER    ( c1 -- c2   Remove parity, control characters )
   127 AND
   DUP 10 <>  OVER 13 <> AND  OVER 08 <> AND
   IF  32 MAX  THEN ;    ( Not LF, CR, or BS )
: MREAD    ( -- c     '-1' returned if none available )
   LPTR @  IOFF PTR @ ION  U<  IF  ( Buffer is not empty )
     LPTR @ C@  FILTER  DUP LPTR @ C!  1 LPTR +!
   ELSE  -1  THEN ;


Scr # 6          A:FTERM.BLK
( Initialize port, serial output - not using interrupt 4-10-86 )
HEX
3FE CONSTANT MSTATUS    ( Modem status register )
3F8 CONSTANT TBUFFER    ( Transmit buffer )
CODE CALL-SERIAL    ( x1 n -- x2 )
   DX POP   AX POP   14 INT   AX PUSH   NEXT  END-CODE
: CALL-PORT1    ( x1 -- x2     Call to Port 1 ) 0 CALL-SERIAL ;
: MINITIALIZE    ( nspeed --      Initialize port )
   ( nspeed = 300 or 1200, no parity, 1 stop bit, 8 data bits )
   12C ( '300' in hex ) =  IF  43  ELSE  83  THEN
   CALL-PORT1  DROP ;    ( Easily extended to 2400 )
: MWRITE    ( c --      Write one character to modem )
   0B MCONTROL PC!    ( DTR, RTS )
   BEGIN MSTATUS PC@ 10 AND UNTIL
   BEGIN  LSTATUS PC@  20 AND  UNTIL  TBUFFER PC! ;
DECIMAL
```

```
Scr # 7        A:FTERM.BLK
( File I/O                                              4-10-86 )
HEX
: FCREATE   ( nattrib asciiz -- nhandle )
   0 ROT  3C00 DOS2 ;
: FOPEN   ( naccess asciiz -- nhandle )
   SWAP 0 0 ROT  3D00 +  DOS2 ;
: FCLOSE   ( nhandle -- nread )
   0 0 ROT  3E00 DOS2 ;
: FREAD   ( adata ndata nhandle -- f )
   3F00 DOS2 ;
: FWRITE   ( adata ndata nhandle -- nwritten )
   4000 DOS2 ;
( Note - should test carry flag - make error arg negative )
: FMOVE   ( dbytes nmethod nhandle -- f   Move file pointer )
   SWAP 4200 +  DOS2 ;
DECIMAL
```

```
Scr # 8        A:FTERM.BLK
( Files - write bytes to new file, or concat. to old   4-10-86 )
2 CONSTANT READ-WRITE   ( Access code: 0=read, 1=write, 2=both )
2 CONSTANT FROM-EOF   ( Method code to move file ptr:
   0 = offset from beginning of file, 1 = from current location,
   2 = end of file + offset )
: FSTART   ( asciiz -- ahandle   Open or create a file )
   READ-WRITE OVER FOPEN
   DUP -2 =  IF ( Not found )  DROP 0 SWAP FCREATE
   ELSE  SWAP DROP  DUP 0 0 ROT  FROM-EOF SWAP  FMOVE DROP
   THEN ;
: FDO   ( ad nd asciiz --   Write new or concatenated )
   FSTART  DUP >R FWRITE DROP R>  FCLOSE DROP ;
```

```
Scr # 9        A:FTERM.BLK
( Files - Default name, and word to change the name    4-10-86 )
CREATE FILE-NAME  66 C, 58 C, 84 C, 69 C, 82 C, 77 C, 0 C,
   74 ALLOT   ( Default name is B:TERM - ASCIZ format )
: GET-STRING   ( a --    ASCIIZ input )
   DUP 80 EXPECT
   SPAN @ 0>  IF  SPAN @ +  0 SWAP C!  ELSE  DROP  THEN ;
: GET-NAME   ( -- )
   CR  ." Name now is: "   FILE-NAME 80 OVER + SWAP  DO
      I C@ 0=  IF  LEAVE  ELSE  I C@ EMIT  THEN  LOOP
   CR  ." New name: "   FILE-NAME GET-STRING CR  ;
```

```
Scr # 10        A:FTERM.BLK
( Terminal program - miscellaneous words               4-10-86 )
VARIABLE FIRST-TEST    TRUE FIRST-TEST !
: FIRST-TIME?   ( -- F )   FIRST-TEST @  FALSE FIRST-TEST ! ;
: ESC-MSG   ( -- )
   CR  ." (S)ave  (N)ame  (K)ill  (U)nkill  (Z)exit"  CR ;
: AT   ." WOOPS! " ;   ( Avoid crash from easy mistake )
VARIABLE BPTR    ( Backup value of pointer - for UNKILL )
VARIABLE BLPTR   ( Backup last value of pointer )
: TSAVE    PAD ;   ( Beginning of terminal buffer )
: TSAVE-END    FIRST ;   ( End of terminal buffer )
```

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.

```
Scr # 11          A:FTERM.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 @  <>  IF  ( Not empty already )
    PTR @ BPTR !   LPTR @ BLPTR !
    TSAVE PTR !  TSAVE LPTR !  THEN  ION ;
: ?AUTOSAVE    ( --    If buffer nearly full, save automatically )
  IOFF PTR @ ION  TSAVE-END 1000 -  U>  IF  CR ." FULL " CR
    19 MWRITE ( XOFF )  10000 0 DO LOOP  ( Delay )
    WRITE-FILE  KILL-FILE
    17 MWRITE ( XON )  THEN ;

Scr # 12          A:FTERM.BLK
( Terminal program - escape functions                   4-10-86 )
: UNKILL-FILE    ( -- )
  IOFF  BPTR @ PTR !  BLPTR @ LPTR !  ION  ;
: ESCAPE-FN    ( c -- f    T=exit )
  DUP 75 ( Kill ) =  IF  KILL-FILE ." Done "  THEN
  DUP 78 ( Name ) =  IF  GET-NAME  THEN
  DUP 83 ( Save ) =  IF  WRITE-FILE KILL-FILE ." Done "  THEN
  DUP 85 ( Unkill ) =  IF  UNKILL-FILE ." Done "  THEN
  DUP 90 ( Zexit ) =  IF  WRITE-FILE  IOFF 0 BDOS  THEN
  DUP 70 ( Forth ) =  IF  DROP 1  ELSE  DROP 0  THEN ;

Scr # 13          A:FTERM.BLK
( Terminal program                                      4-10-86 )
: TERM    ( --      Dumb terminal program )
  CR  ." Use ESC for help" CR
  FIRST-TIME? IF  1200 MINITIALIZE   TSAVE PTR !  TSAVE LPTR !
    ISETUP ION  THEN
  FALSE ( Loop control - set to TRUE to exit )
  BEGIN    ?AUTOSAVE
    MREAD     ( Read one character )
    DUP -1 =  IF  DROP  ELSE  EMIT  THEN
    KEY? IF  ( If key typed, send 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
    THEN
  DUP UNTIL  DROP ;

Scr # 14          A:FTERM.BLK
( Finishup                                              4-10-86 )
( Optional, F83 only - sets automatic execution )
: HELLO2   ONLY FORTH ALSO DEFINITIONS   TERM ;
' HELLO2 IS BOOT

( SAVE-SYSTEM TERM.COM )
```

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 **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 **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

$$(1)\quad \sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \dots$$

$$(2)\quad \sin x \approx x(1 - \frac{x^2}{6}(1 - \frac{x^2}{20}(1 - \frac{x^2}{42}(1 - \frac{x^2}{72}))))$$

$$(3)\quad \sin x \approx \frac{y}{K}(1 - \frac{y^2}{6K^2}(1 - \frac{y^2}{20K^2}(1 - \frac{y^2}{42K^2}(1 - \frac{y^2}{72K}))))$$

$$(4)\quad z = \frac{K_1}{K_2} X \quad ; \quad \frac{K_1}{K_2} >> 1$$

$$(5)\quad \sin x \approx \frac{K_2}{K_1} z (1 - (\frac{K_2}{K_1})^2 \frac{z^2}{6}(1 - (\frac{K_2}{K_1}) \frac{z^2}{20}(1 - (\frac{K_2}{K_1}) \frac{z^2}{42}(1 - \frac{K_2}{K_1} \frac{z^2}{72}))))$$

$$(6)\quad xs = \frac{z * z}{k_1}$$

$$(7)\quad \frac{K_1}{K_2} \sin x = \frac{z}{K_1}(K_1 - \frac{K_2^2}{6} \frac{XS}{K_1}(K_1 - \frac{K_2^2}{20} \frac{XS}{K_1}(K_1 - \frac{K_2^2}{42} \frac{XS}{K_1}(K_1 - \frac{K_2^2 XS}{72}))))$$

$$(8)\quad 3784 \sin x = \frac{z}{2^{16}}(2^{16} - \frac{50\ XS}{2^{16}}(2^{16} - \frac{15\ XS}{2^{16}}(2^{16} - \frac{7.14\ XS}{2^{16}}(2^{16} - 4.2\ XS))))$$

$$(9)\quad \frac{1}{\text{scaling factor}} = \frac{1}{K_1/K_2} = \frac{1}{3784} = 0.026\ \%$$

$$(10)\quad a\ b\ \text{--}\ m \text{ with } m = 2^{16} - \frac{a * x^2 * b}{2^{16}}$$

(Continued on page 19)

FORTH INTEREST GROUP MAIL ORDER FORM

P.O. Box 8231 San Jose, CA 95155 (408) 277-0668

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

200 - ALL ABOUT FORTH .... $25/26/35 ______
Glen B. Haydon
An annotated glossary for MVP Forth; a 79-Standard Forth.

205 - BEGINNING FORTH .... $17/18/21 ______
Paul Chirlian
Introductory text for 79-Standard.

215 - COMPLETE FORTH .... $16/17/20 ______
Alan Winfield
A comprehensive introduction including problems with answers. (Forth 79)

220 - FORTH ENCYCLOPEDIA .... $25/26/35 ______
Mitch Derick & Linda Baker
A detailed look at each fig-Forth instruction.

225 - FORTH FUNDAMENTALS, V. 1 .... $16/17/20 ______
Kevin McCabe
A textbook approach to 79-Standard Forth.

230 - FORTH FUNDAMENTALS, V. 2 .... $13/14/16 ______
Kevin McCabe
A glossary.

232 - FORTH NOTEBOOK .... $25/26/35 ______
Dr. C. H. Ting
Good examples and applications. Great learning aid. PolyFORTH is the dialect used. Some conversion advice is included. Code is well documented.

233 - FORTH TOOLS .... $20/21/24 ______
Gary Feierbach & Paul Thomas
The standard tools required to create and debug Forth-based applications.

235 - INSIDE F83 .... $25/26/35 ______
Dr. C. H. Ting
Invaluable for those using F-83.

237 - LEARNING FORTH .... $17/18/21 ______
Margaret A. Armstrong
Interactive text, introduction to the basic concepts of Forth. Includes section on how to teach children Forth.

240 - MASTERING FORTH .... $18/19/22 ______
Anita Anderson & Martin Tracy
A step-by-step tutorial including each of the commands of the Forth-83 International Standard; with utilities, extensions and numerous examples.

245 - STARTING FORTH (soft cover) .... $20/21/24 ______
Leo Brodie
A lively and highly readable introduction with exercises.

246 - STARTING FORTH (hard cover) .... $24/25/29 ______
Leo Brodie

255 - THINKING FORTH (soft cover) .... $16/17/20 ______
Leo Brodie
The sequel to "Starting Forth". An intermediate text on style and form.

265 - THREADED INTERPRETIVE LANGUAGES $23/25/28 ______
R.G. Loeliger
Step-by-step development of a non-standard Z-80 Forth.

270 - UNDERSTANDING FORTH .... $3.50/5/6 ______
Joseph Reymann
A brief introduction to Forth and overview of its structure.

FORML CONFERENCE PROCEEDINGS

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

310 - FORML PROCEEDINGS 1980 .... $30/33/40 ______
Technical papers on the Forth language and extensions.

311 - FORML PROCEEDINGS 1981 (2V) ...... $45/48/50 ______
Nucleus layer, interactive layer, extensible layer, metacompilation, system development, file systems, other languages, other operating systems, applications and abstracts without papers.

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

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

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

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

ROCHESTER PROCEEDINGS

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

321 - ROCHESTER 1981 (Standards Conference) $25/28/35 ______
79-Standard, implementing Forth, data structures, vocabularies, applications and working group reports.

322 - ROCHESTER 1982
(Data bases & Process Control) .... $25/28/35 ______
Machine independence, project management, data structures, mathematics and working group reports.

323 - ROCHESTER 1983 (Forth Applications) . $25/28/35 ______
Forth in robotics, graphics, high-speed data acquisition, real-time problems, file management, Forth-like languages, new techniques for implementing Forth and working group reports.

324 - ROCHESTER 1984 (Forth Applications) . $25/28/35 ______
Forth in image analysis, operating systems, Forth chips, functional programming, real-time applications, cross-compilation, multi-tasking, new techniques and working group reports.

325 - ROCHESTER 1985
(Software Management and Engineering) $20/21/24 ______
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
2nd edition, Sept. 1984 .... $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 - HANDY REFERENCE CARD .... FREE ______

683 - FORTH-83 HANDY REFERENCE CARD ...... FREE ______

FORTH MODEL LIBRARY

The model applications disks described below are new additions to the Forth Interest Group's library. These disks are the first releases of new professionally developed Forth applications disks. Prepared on 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 PC/FORTH 3.0
MasterFORTH 1.0	TaskFORTH 1.0
PolyFORTH®II	

Forth-83 Compatibility Macintosh
MasterFORTH

ORDERING INFORMATION

701 - A FORTH LIST HANDLER V.1 .... $40/43/45 ______
by Martin J. Tracy
Forth is extended with list primitives to provide a flexible high-speed environment for artificial intelligence. ELISA and Winston & Horn's micro-LISP are included as examples. Documentation is included on the disk.

702 - A FORTH SPREADSHEET V.2 .... $40/43/45 ______
by Craig A. Lindley
This model spreadsheet first appeared in Forth Dimensions Volume 7, Issue 1 and 2. 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

Name ____________________

Member Number ____________________

Company ____________________

Address ____________________

City ____________________

State/Prov. ____________ ZIP ____________

Country ____________________

Phone ____________________

OFFICE USE ONLY

By ________ Date ________ Type ________

Shipped By ________ Date ________

UPS Wt. ________ Amt. ________

USPS Wt. ________ Amt. ________

BO Date ________ Wt. ________ Amt. ________

By ________

ITEM #	TITLE	AUTHOR	QTY	UNIT PRICE	TOTAL
108	MEMBERSHIP →				SEE BELOW

SUBTOTAL	
10% MEMBER DISCOUNT MEMBER #	
SUBTOTAL	
CA. RESIDENTS SALES TAX	
HANDLING FEE	**$2.00**
MEMBERSHIP FEE ☐ NEW ☐ RENEWAL $30/37/43	
TOTAL	

☐ Check enclosed (payable to: **FORTH INTEREST GROUP**)

☐ VISA ☐ MASTERCARD

Card # ____________________

Expiration Date ____________________

Signature ____________________

($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. $15 minimum on charge orders. Checks must be in US$, drawn on a US Bank. A $10 charge will be added for returned checks.

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

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

SALES TAX
Deliveries to Alameda, Contra Costa, San Mateo, Los Angeles, Santa Cruz Counties, add 6½%; Santa Clara County, add 7%; other California Counties, add 6%.

(Continued from page 14)
factor k in order to prevent overflow. Here we choose y = kx and rewrite (2) as in (3).

For the reasons explained later on, we shall actually use two scaling factors (4) such that (2) becomes (5).

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:

```
z z U* SWAP DROP
```

which is the same as

z z k_1 */with $k_1 = 2^{16}$ but much faster.

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 choosen 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 **SIN1** which evaluates the sine of values between zero and 5994 (3784 * PI/2 = 5944), corresponding to zero and 90°. The series may only be evaluated for arguments greater than 256 (= 4^0), 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

$$\frac{z * z}{2^{16}}$$

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

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

DEG and **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 **SIN1**. **SIN1, COS1** and **TAN1** are about twice as fast as **SIN, COS** and **TAN** but accept input only in the range of 0 < z < 5944 (angles between 0 and 90°), whereas **SIN, COS** and **TAN** accept any input between -2^{15} and $+2^{15} - 1$.

```
30 DEG SIN1 KTIMES — 500
90 DEG SIN1 KTIMES — 1000
```

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

Index to Advertisers

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**).

```
           02 VARIABLE c      ( invert keyboard case as default        )

: UC         1 c  ! ;         ( set case conversion to lower/uppper )

: LC         0 c  ! ;         ( set no case conversion                 )

: IC         2 c  ! ;         ( set case conversion to invert case    )

: KEY         ( --- key                                                )
  (KEY)                       ( GET NEXT KEYSTROKE                     )
  c @                         ( ANY TRANSLATION?                       )
  IF DUP 060 > OVER 07B < AND ( CHECK IF LOWER CASE                    )
  c @ 1- IF                   ( c=2? IS CASE INVERSION                 )
  OVER DUP 040 > SWAP 05B < AND ( CHECK IF UPPER CASE                  )
  OR IF                       ( UPPER OR LOWER CASE LETTER?            )
  BL XOR THEN                 ( YES, INVERT CASE                       )
  ELSE IF                     ( c=1, LOWER CASE?                       )
  BL -                        ( YES, CONVERT TO UPPER                  )
  THEN THEN THEN ;

: ?TERMINAL                   ( --- flag                               )
  FALSE (?KEY)                ( KEYSTROKE WAITING?                     )
  IF KEY 3 =                  ( YES, GET IT, IS IT CNTL-C?             )
  IF DROP TRUE                ( YES, RETURN TRUE                       )
  ELSE KEY 3 =                ( NO, WAIT FOR NEXT KEY, =CNTL-C?        )
  IF DROP TRUE                ( YES, RETURN TRUE                       )
  THEN THEN THEN ;

: -->                         ( ---                                    )
  ?LOADING ?TERMINAL          ( CHECK FOR CNTL-C                       )
  IF QUIT THEN                ( YES, QUIT LOADING                      )
  0 >IN !                     ( RESET >IN                              )
  B/SCR BLK @ OVER MOD        ( # BUFFERS IN BLOCK PARSED SO FAR       )
  - BLK +! ; IMMEDIATE        ( INC BLK BY AMOUNT LEFT                 )
```

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 **SELECT** and **ORDERED**).

A glossary for these and the relevant Simple File words follows.

SELECT (-- <fieldname> <string>)
selects from the file all records where there is a match between the given field and the given string, and builds an array of locators to them.

ORDERED (-- <fieldname>)
puts the array of locators in alphabetic sequence by fieldname. A **SELECT** must have been done some time previously. Once a **SELECT** has been made, different **ORDERED**s may be done to it.

PERFORM (— <words for interpretation>)
issues a **CR** and interprets the rest of the input stream for each selected record.

FIELD< (locator1, locator2 -- f)
reads the indicated records as needed and compares the <fieldname> given in the command, **ORDERED** <fieldname>.

SELECTED (n -- addr)
is the array of selected locators.

TALLY (-- addr)
is a variable that contains the number of selected records.

#RECORD (-- addr)
is a variable that points to the current word.

-FIND (-- f)
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.

```
: ARRAY  CREATE  2* ALLOT DOES> OVER + + ;  ( you may already have this )

MAXRECS ARRAY SELECTED

VARIABLE TALLY       0 TALLY !

: SELECT  ( --<fieldname> <string> )
    '  KEEP  TOP  0
    BEGIN -FIND ( this is simple files '-FIND' ) NOT
    WHILE  RECORD C@ BL >
        IF #RECORD @ OVER SELECTED ! 1+ THEN
    REPEAT
    TALLY ! ;

: FIELD<  ( locator1,locator2 -- f )
    #RECORD !  KIND @ FIELD  WHAT SWAP CMOVE
    #RECORD !  KIND @ FIELD  WHAT
    -TEXT 0< ;

: ORDERED  ( --<fieldname> )
    '  KIND !  0 SELECTED  TALLY @  SORTED FIELD< ;

: PERFORM  ( --<words to be executed> )
    TALLY @ ?DUP
    IF  >IN @ >R  0
       DO  CR  I SELECTED @  #RECORD !
           R> R> R@ >IN ! >R >R   INTERPRET
       LOOP
    THEN
    R> DROP ;
```

TI 99/4A ISR Installation

Gene Thomas
Little Rock, Arkansas

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**, **DDOT** and **SMASH** terminate with **NEXT**.) Because the ISR executes in sixteen milliseconds or less and is called so often, it appears to the operator that this background task is both simultaneous 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**, **GRAPHICS2**, **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 0ISR VDPMDE @ ...

Screen 22, line 11

... CLS SCRNO 1ISR DROP ...

Screen 33, line 15

DECIMAL "XX" CLOAD 0ISR R->BASE

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

Screen 38, line 1

: VED 0ISR BOX SWAP ...

Screen 38, line 2

OF OF 1ISR BCK ...

Screen 51, line 4

: TEXT 0ISR ...

Screen 52, line 4

: GRAPHICS 0ISR ...

Screen 53, line 4

: MULTI 0ISR ...

Screen 54, line 3

: GRAPHICS2 0ISR 0A0 1 VWTR ...

Screen 3, before the final **R->BASE:**

INSTALL-ISR BAS?

Here is how the ISR is accomplished (refer to screen 41). Lines 0 - 4 are self-explanatory. The colon word **BAS?** is the task that the ISR will cause to be executed. The conversions needed to accomplish screen writes through **U.**, **.**, **EMIT**, etc. are all much too slow for ISR execution, so we must depend on the directness of **VMBW** (video multiple byte write), which calls on machine code. We go one step further and eliminate two dictionary searches by using **SYSTEM**. **VMBW** is defined as:

: VMBW 2 SYSTEM ;

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 radix buffer **BAS** . That conversion is accomplished on line 5. For example,

16 10 /MOD . . 1 6 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 digits. 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 buf for vdp write)
  3 0 VARIABLE STR\ 4 ALLOT  ( string buf for vdp write)
  4 STR\  !" Base="  ( 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 @ 30 + ( read from/write to addr's {radix})
 10      2 2 SYSTEM ( write 2 bytes {##})  ;
 11 HEX
 12 : 0ISR  ( -- ;p ISR off)  0 83C4 !
 13 : 1ISR  ( -- ;p ISR on )  INTLNK @ 83C4 ! ;
 14 : INSTALL-ISR  ( -- ;p followed by WORD to be installed)
 15    0ISR  [COMPILE] ' BL WORD CFA ISR ! ;                   R->BASE
```

variable which contains the CFA of the ISR to be executed. (The TI-99/4A executes on CFAs, and not on PFAs as do some systems.) **INTLNK** 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 BAS?   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 a no-op word. There is just such a resident word, **NOP**. **NOP** is defined as **: NOP ;**. Ergo, **INSTALL-ISR NOP 1ISR** is the best way. You may want to re-define **INSTALL-ISR** as:

```
: INSTALL-ISR   0ISR [COMPILE]
' BL WORD CFA ISR !   1ISR ;
```

The commands would then be **INSTALL-ISR BAS?** and **INSTALL-ISR NOP**. By the way, the **[COMPILE]** is necessary because **'** is an immediate word and would otherwise execute during compilation rather than being compiled as desired.

Here are a few things to keep in mind as you write and install your own ISRs.

1. It must execute in sixteen milliseconds or less. Otherwise, there will be too little time left in each second for meaningful computing.

2. *Do not* call an ISR within an ISR.

3. ISRs must neither expect nor leave anything on the stack. They may use the stack internally, however.

4. All ISR activity stops during calls to DSRs (device service routines), then resumes (as in disk read/write).

5. Always turn off user ISR activity with **0 83C4 !** before installing a different ISR. To do otherwise is to invite a fatal error resulting in a system crash. The same applies to pushing the CFA into ISR before turning it on with **INTLNK @ 83C4 !**.

How do you know if the routines' execution time is sixteen milliseconds or less? 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**, **VSBW**, **VSBR**, **VMBR**, **AND**, **OR**, **XOR**, **/MOD**, **U/**, ***/**, **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 @** in **BAS?**. I suggest 31 and 38 respectively, to put the display in the upper right corner.

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 drawing on Chuck's unique perspective on the relationship between men and computers, as well as specific questions about the new native 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 reading to 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 running directly off a terminal.

(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.

(George M.) What is your opinion of "fast" Forths (with, for instance, sixteen threads) that have the feel of packaged languages, as opposed to simpler fig-FORTH types?

(Chuck) The chip is so fast that compile is instantaneous. It recompiles itself over a 9600-baud line in thirty seconds (thirty blocks). This with two threads. But, to answer the question, compile time is important — the style of Forth code depends upon reaction time. Slow compiles lead to vastly different attitudes.

(David B.) Back to software implementations of Forth... Is it time for a new standard? Current standards (Forth-79, Forth-83) don't know thirty-two bits, have no files, primitive I/O, etc. What's the future for Forth? Are we ready for the mainstream?

(Chuck) Standards are a red flag. If I had the slightest concern for them, Forth nor chip could not be. Standards are for communication among humans, not computers. Files, I/O are horribly hardware dependent. Blocks are universal. Chip I/O is strongly application dependent. For every problem are many solutions. Why choose [only] one?

(David B.) But we all seem to reinvent the wheel: number input, for example. Maybe just a standard user interface of some kind.

(Chuck) The wheel has been reinvented many times. It will be. To keep it simple, the overhead must be brutally pruned. I do this constantly, and I regret the inefficiency. But the most standard language must be Ada, and its efficiency is zero.

(David Butler) Is it important for Forth to become more popular, or will it always be an "elitist" (best possible light) language for power-programmers?

(Chuck) I've said I don't object to Forth being elitist. The widest possible acceptance would be nice, but not at a compromise of principle.

(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!

(Kiyoshi Y.) 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 useable, 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.

(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) Do you expect to see a Novix development board more within the price range of the hobbyist, and will there be support for that sector? $900 is a bit steep for the average hacker (that would buy quite a few 65F12's, for example).

(Chuck) Sweet words. I wish my colleagues could hear them. The Gamma board was priced at $400 as a kit. That is not profitable. It may be available in some form, from someone. The problem is support. No one can afford Heathkit-type documentation at that price. But a board, chip and PROMs could be available, if demand warrants.

(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.

(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 Fortran 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 groundswell 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

Tucson Chapter
Twice Monthly,
2nd & 4th Sun., 2 p.m.
Flexible Hybrid Systems
2030 E. Broadway #206
Call John C. Mead
602/323-9763

• ARKANSAS

Central Arkansas Chapter
Twice Monthly, 2nd Sat., 2p.m. & 4th Wed., 7 p.m.
Call Gary Smith
501/227-7817

• CALIFORNIA

Los Angeles Chapter
Monthly, 4th Sat., 10 a.m.
Hawthorne Public Library
12700 S. Grevillea Ave.
Call Phillip Wasson
213/649-1428

Monterey/Salinas Chapter
Call Bud Devins
408/633-3253

Orange County Chapter
Monthly, 4th Wed., 7 p.m.
Fullerton Savings
Talbert & Brookhurst

Fountain Valley
Monthly, 1st Wed., 7 p.m.
Mercury Savings
Beach Blvd. & Eddington
Huntington Beach
Call Noshir Jesung
714/842-3032

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

Sacramento Chapter
Monthly, 4th Wed., 7 p.m.
1798-59th St., Room A
Call Tom Ghormley
916/444-7775

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

• 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

Tampa Bay Chapter
Monthly, 1st. Wed., p.m.
Call Terry McNay
813/725-1245

• 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
312/879-3242

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

Fort Wayne Chapter
Monthly, 2nd Tues., 7 p.m.
IPFW Campus
Rm. 138, Neff Hall
Call Blair MacDermid
219/749-2042

• 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

Fairfield FIG Chapter
Monthly, 4th day, 8:15 p.m.
Call Gurdy Leete
515/472-7077

• KANSAS

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

• 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
313/562-8506

• MINNESOTA

MNFIG Chapter
Even 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

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

• NEVADA

Southern Nevada Chapter
Call Gerald Hasty
702/452-3368

• NEW HAMPSHIRE

New Hampshire Chapter
Monthly, 1st Mon., 6 p.m.
Armtec Industries
Shepard 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

Syracuse 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

Cincinatti 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., 1 p.m.
Tektronix Industrial Park
Bldg. 50, Beaverton
Call Tom Almy
503/692-2811

• **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 Tue., 7:30 p.m.
Sci. Appl. Int'l. Corp., 8th Fl.
800 Oak Ridge Turnpike, Oak Ridge
Call Richard Secrist
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 VanSyckel
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 Shprentz
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., 8 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
Lariksdreff 20
2120 Schoten
03/658-6343

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

• **CANADA**

Alberta Chapter
Call Tony Van Muyden
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. Solntseff
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.,
7p.m., Rm. 408
Polytechnic of South Bank
Borough Rd., London
D.J. Neale
58 Woodland Way
Morden, Surry SM4 4DS

• **FRANCE**

French Language Chapter
Contact Jean-Daniel Dodin
77 Rue du Cagire
31100 Toulouse
(16-61)44.03.06

• **GERMANY**

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

• **HOLLAND**

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

FIG des Alpes Chapter
Contact: Georges Seibel
19 Rue des Hirondelles
74000Annely
50 57 0280

• **IRELAND**

Irish Chapter
Contact Hugh Doggs
Newton School
Waterford
051/75757 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
812-2111 ext. 7073

• **NORWAY**

Bergen Chapter
Kjell Birger Faeraas
Hallskaret 28
Ulset
+47-5-187784

• **REPUBLIC OF CHINA**

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

• **SWEDEN**

Swedish Chapter
Hans Lindstrom
Gothenburg
+46-31-166794

• **SWITZERLAND**

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

SPECIAL GROUPS

Apple Corps Forth Users Chapter
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
408/425-8700

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

BULK RATE
U.S. POSTAGE
PAID
Permit No. 3107
San Jose, CA

Address Correction Requested