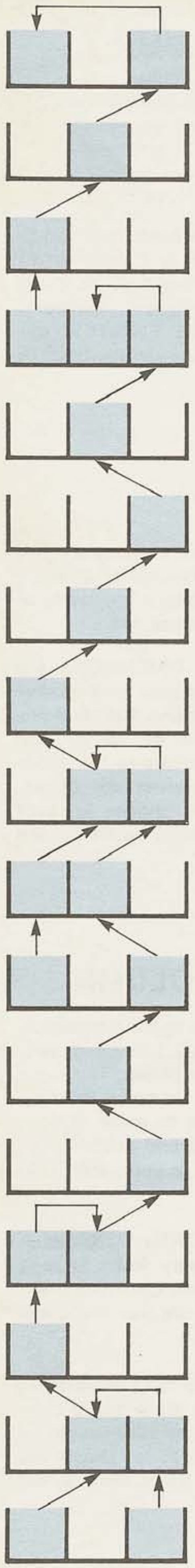
FORTH DIMENSIONS

FORTH INTEREST GROUP
P.O. Box 1105
San Carlos, CA 94070

Volume III
Number 2
Price $2.50



INSIDE

FORTH DIMENSIONS

Published by Forth Interest Group

Volume III No. 2 July/August 1981

Publisher Roy C. Martens
Editor C. J. Street

Subscription to FORTH DIMENSIONS is free with membership in the Forth Interest Group at $15.00 per year ($27.00 foreign air). For membership, change of address and/or to submit material, the address is:

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

HISTORICAL PERSPECTIVE

FORTH was created by Mr. Charles H. Moore in 1969 at the National Radio Astronomy Observatory, Charlottesville, VA. It was created out of dissatisfaction with available programming tools, especially for observatory automation.

Mr. Moore and several associates formed FORTH, Inc. in 1973 for the purpose of licensing and support of the FORTH Operating System and Programming Language, and to supply application programming to meet customers' unique requirements.

The Forth Interest Group is centered in Northern California. Our membership is over 2,400 worldwide. It was formed in 1978 by FORTH programmers to encourage use of the language by the interchange of ideas through seminars and publications.

EDITOR'S COLUMN

The feedback on our new applications editorial policy is all positive. To date, we are receiving a nice variety of articles. I want to urge our members not to slack off. In order to keep up a steady flow of quality output, we need quality input—IN QUANTITY.

If you have an article you have been meaning to write, please get it down and send it in. If you havr an application, programming trick or tool that you have found useful, please share it with our members. Remember: YOU DON'T HAVE TO BE A WRITER—our staff is set up to help you with whatever you need to make your idea publishable.

Please send all submissions to:

Editor
FORTH DIMENSIONS
P.O. Box 1105
San Carlos, CA 94070

HEX is back this month, and there are photos of the Rochester Conference courtesy of George W. Shaw, II. We are always looking for photos (black and white or color prints preferred) and cartoon ideas, too.

Starting next edition, FORTH DIMENSIONS will have a marketing column in a question and answer format. If you have had ideas, programs, etc., that you wondered how to sell, this column will be for you. Please direct your marketing questions to the above address. Questions of general interest will be answered in this column by experts chosen for their knowledge of marketing and computer hardware and software.

C. J. Street
Editor

PUBLISHER'S COLUMN

Lots of good news! The reaction to the application orientation of FORTH DIMENSIONS has been very positive. Thanks to our editor, Carl Street. The more articles you send Carl, the closer we come to being able to go monthly. Our plans are to make FORTH DIMENSIONS more general interest and publish high level (sic) technical material twice a year, ala, 1980 FORML Proceedings.

Plans for the 1981 FORML (FORTH Modification Laboratory) Conference are underway. Refer to page 63 for more details. The FIG National Convention will be on Saturday, November 28th in the San Francisco Bay Area. Make your plans.

Now, some bad news! We have to raise some of our prices. It's been a couple of years since we've done any price adjusting and cost increases have caught up with us. The order form on the last page reflects the new costs which are now in effect Sorry, we'll do our best to hold the line.

Roy Martens

LETTERS

Dear FIG,

Please find enclosed two short articles which might be suitable for publication in FORTH DIMENSIONS.

I did not ask for the publication kit, so I hope the articles do not violate your rules too much. Second, my native language is not English but Dutch, so forgive me if there are any errors and feel free to correct them.

Please note my new telephone number and correct it in your listing of local FIG chapters.

We have not had many meetings lately, probably because our members are too active!

Paul van der Eijk
5480 Wisconsin Avenue #1128
Chevy Chase, MD 20015
(301) 656-2772

Thank you for your articles, Paul. Readers can find them under the applications area of this issue.--ed.

Dear FIG,

I recently purchased a listing of fig-FORTH for the 8080 from you and I am very impressed with the Language package. You will find enclosed an order and Bank Draft for several books which I eagerly await. I received my Dual Micropolis Mod II Disk drives only two weeks ago and my first project was to assemble FORTH. The disk interface routines were quite easy to link to the Micropolis DOS using ideas from the CPM interface supplied. However, when I tried to LOAD a short word definition off a screen the system would lock up and not come back with any error messages or the 'OK'; because the system would compile words from the keyboard and the Disk I/O worked well, I was puzzled as to why there was a problem. After four days of searching and debugging, I found that the program was looping through INTERPRET, and each time the parameter stack had an extra value on it. Eventually, I found the bug; it was in the ENCLOSE routine and the problem is that only an 8 bit counter is used to hold the offset into the buffer. However, the Micropolis sectors are 256 bytes and so are my Forth Disc Buffers. If there are any non-delimiter characters in a buffer, then all works OK. However, if the buffer holds 256 blanks, then the loop around ENCL1 scans to the end of the buffer but the 8 Bit offset ends up pointing at the start of the buffer still an INTERPRET never gets to to see the NULL at the end of the buffer. Obviously, the routine works OK for CPM 128 Byte Sectors, but needs modifying for larger capacity sectors.

I have included the source listing for ENCLOSE as modified by me (sorry, I haven't got my printer going yet). I have used the DE register pair for the offset counter and have kept the definition character in the Accumulator which means pushing and popping it when it is necessary to check for a NULL.

I hope you find this of interest and maybe you will include a change of this sort in future versions. I learned a great deal from this problem, and it was probably to my advantage that it occurred, as my only prior information was the 'FORTH' BYTE. I really learned the hard way.

William D. Miles
P. O. Box 225
Red Cliffs
Victoria, 3496
Australia

Thank you for your contribution. NOTE: You will find Mr. Miles' bug fix in the TECHNOTES, BUGS & FIXES section of this issue.--ed.

DON'T MISS OUT! GET YOUR PAPER IN EARLY FOR THE FORML CONFERENCE!

Dear FIG,

Could you print my address in your next FORTH DIMENSIONS issue: I would like to hear from other Belgian FORTH-ists!

Michel Dessaintes
Rue de Zualart 64
B 5810 Suarlee
Begium

OK, Michel, start watching your mail box!--ed.

Dear FIG,

Congratulations on your last issue (Vol. II, No. 6). It's nice to see some tutorial inputs at a level that beginners like me can understand. Keep it up!

Would you please print the SEARCH routine mentioned in John James' article on page 165 of Vol. II, No. 6. It apparently got replaced by the correction notice at the bottom of the page.

I was interested in trying EDGAR H. FEY'S FEDIT in Vol. II, No. 5, but was stumped by the word REPL which was not defined. Is it possible MR. FEY could provide the definition? (Also, I noted that SCR#67 errors at line 48 -B/BUD -- which apparently is supposed to be B/BUF.) Screens should be required to be loadable, not edited by publisher or author without loading edited version.

In respect to editing, please also note that Major Selzer's article in the Vol. II, NO. 3 issue on page 83, SCR#200 line 8 should apparently be 08 CASE for left cursor as opposed to OB as printed, since OB is used for UP cursor. This screen does work when above mentioned change is made.

I realize that submitted copy may need to be retyped but the dangers of introducing errors are ever present. I'm sure that you catch most of them.

Robert I. Demrow
P. O. Box 158 BLUE STA.
Andover, MA 01810

Thank you for your thoughts. Glad you like our new approach. John James SEARCH is in a previous issue. Regarding errors, we do try to minimize them; but we are only human.--ed.

Dear FIG,

During September of 1980, material was ordered which included hard copy of figFORTH for the Motorola 6802 (6809 preferred) CPU, and FIG membership for a year. Hard copy received was Talbot Microsystems v.1.1. 6809 FORTH. After a considerable amount of study, and a complete rewrite, that software is now up and running, apparently as designed. (No operating bugs have been detected, but it would be reasonable to expect bugs to appear far into the future.) Some general comments on the system may be of interest.

A major factor in the acquistion of this software was the indicated ability to run high level software on a small system. If the Talbot software is designed for a microsystem, then I must have a nano-system by definition; a disk would cost far more than all hardware currently in use, and appears quite unrealistic for this home hobby system. The alternative cassette is implemented, but patience would be strained beyond limits if nearly 8K words were loaded for each use at 300 baud. Thus, my system clearly demands use of EPROMs for source code.

I have used several methods of code reduction.

1. A short branch to several copies of NEXT.

2. Place the user area in the direct pad.

3. Add a byte literal as well as LIT.

4. Some high level routines are shorter in code.

The end result of this process was code retaining nearly all of Mr. Talbot's word definitions, and fitting easily in 6K bytes (3 2716's). There is very little benchmark information available (this would make a worthwhile FORTH DIMENSIONS article), but those found generally ran in 1/2 the time cited for the APPLE.

A. R. Gunion
182 Minuteman Drive
Concord, MA

The real definitions of nano and micro as applied to systems vary with each user. Suffice to say that FORTH is by definition a disk based system. If you do not have a disk then you are compromising on an area vital to obtaining the real potential of the system.

Regarding benchmarking, it has always been the position of FORTH DIMENSIONS that the nature of FORTH makes benchmarks more of an indication of the speed of a CPU than any particular system and we generally do not publish them. This has been discussed at length in previous editions.--ed.

Dear FIG,

While I cannot disagree with the intent of "An Open Response" in FORTH DIMENSIONS, Vol. II, No. 6, concerning the hardware requirements for FORTH, I feel you may discourage some with the categorical statements you made. It is possible to accomplish a great deal with much less than you described. I hand-installed the 6502-verison of fig-FORTH on a homebrew, KIM-based system that had only 8K of RAM and traditional cassette-storage. My "terminal" was a memory-mapped 16-line by 32-character display with ASCII keyboard. This minimal system has given me hours of pleasure and practical experience with FORTH, and because of the concise nature of FORTH has been capable of powerful constructs. An acquaintance has installed a cut-down version on a 5K KIM with ASCII keyboard and walking "times-square" display on the KIM LED's. There is no question that we would be more comfortable in the hardware environment you define, but compared to Tiny-Basic, for example, these minimal FORTH's are heaven.

I found the same bugs in the May 1980 6502-version of fig-FORTH that Grotke and McCarthy have already reported. In addition, I would warn prospective installers that the TRACE routine depends on the output routines preserving the Y-register, and that the MON routine is not quite correct. Since the 6502-processor increments the program counter by two when BRK instruction is executed, BRK should be followed by a NOP to ensure that a simple machine-language monitor will return to the start of the IDX XSAVE instruction.

My system now includes a 320x200 dot raster-scan display, and I am interested in corresponding with others concerning FORTH-based graphics processors.

Kent A. Reed
49 Midline Court
Gaithersburg, MD 20760

The point of the "Open Response" was not to condemn anyone's system; rather to point out that FORTH is designed to be used with a disk. Naturally, the nature of FORTH means that it will perform (and outperform other languages) regardless of the environment. Your "bug" comments are appreciated.--ed.

Dear FIG,

In bringing up the 6502 Assembly Source listing on my Rockwell System 65, I encountered a problem involving writing or reading the disk drives. The symptoms involved setting an 01 error everytime the disk was asked to jump to the next track.

The problem turns out to be hardware and only exists on a Sys 65 with Pertec model FD200 drives. The fix is simple and is detailed in Rockwell Service Bulletin 'SYSTEM 65-7' which may be obtained by writing:

Rodger Doerr
SYSTEM 65 Customer Service Dept.
ROCKWELL INTERNATIONAL
Microelectronic Devices
P. O. Box 3669
Anaheim, CA 92803

(Or call Rodger at (714) 632-2862.)

I hope that this information can be helpful to other individuals who are working with FORTH on the SYS 65.

Jack Haller
230 Mechanic St.
Boonton, NJ 07005

Thank you--I am sure you have saved more than one frustrated programmer a few sleepless nights.--ed.

Dear FIG,

Enclosed is $12.00 (now $15.00--Pub.) for another year of FORTH DIMENSIONS. I have FORTH up on 2 KIM's (Dean's version) and a Superbrain; although my "playtime" is limited, I enjoy tinkering very much. It might amuse you and Mr. Moore to know that one of the systems is going to control a 10' dish radiotelescope which I also use for looking at thunderstorms.

I am slowly getting together parts of a Western Digital-based computer. Their p-code chip is a natural for FORTH--almost all primitives are single instructions. This is a very long-term project and, no doubt, someone will beat me to it, but it needs doing. Please pass this on to anyone who might be interested. I would be glad to correspond with them.

As a long-time but not prolific user of FORTH, I'd like to put in my buck's (inflated two bits') worth: KISS--this acronym is keep it simple, stupid. In other words, let's not get too many words into "Basic FORTH" vocabulary. Certainly, more advanced words are useful and should be published and documented, and are, of course, part of the FORTH vocabulary by definition. Any standards, however, should be kept very simple. Enough.

Don Latham
Six Mile Road
Huson, MT 59846

OK interested members, drop him a line.--ed.

Dear FIG,

This letter is in response to C. A. McCarthy's letter in FORTH DIMENSIONS, Vol. II No. 6 concerning the errors he listed:

Page 0061

Yes, there should be a SEMIS at the end of the UPDATE.

Page 0064

I haven't hooked up disks to FORTH yet, so I didn't notice this one, but I agree that the displacement in line 3075 is wrong.

Page 0067

I dropped one of the STX XSAVE's without ill effect.

Page 0069

The extra SEMIS is superfluous, but will not have any harmful effect.

I did find another error in the listing. This one, rather than being a typo, appears to be an error in program logic.

Page 0017, lines 0803-0805. The listing for routine ZERO shows:

```
LDA 0,X
ORA 1,X
STY 1,X
```

Since Y contains 0 at this point, the zero flag in the processor status register will always be set by the STY instruction. Therefore, the branch which follows will never be taken, resulting in a logical "false" value always being left on top of the stack. I replaced the above code with the following:

```
LDA 1,X
STY 1,X
ORA 0,X
```

This causes the processor status to be set properly to indicate whether the top stack entry is a zero or not. I know of no other errors in the listing.

Steve Wheeler
504 Elmira
Aurora, CO 80010

Thank you for passing along the above.--ed.

Dear FIG,

A little note about changes in the situation in NW Europe. During the second half of March, there was an exhibition in Malmo (close to Copenhagen) - "Datacraft 81" (Computer power-81).

Up until then, FORTH was very difficult to get in touch with here in Sweden. To my great astonishment, there were at least 4, perhaps 6, systems running in different pools. The most interesting one was a poly-FORTH system running on an ABC-80 (a Sw Z-80 lowend machine). There were also fig-FORTH's running realtime setups on PET's.

To me, who had up until then been 'dry-swimming ' FORTH, it was quite an experience to key in definitions, clear, compact, and (CR/LF), to be able to use them. Quite a kick!

Calle Hogard

Glad to hear things are moving ahead.--ed.

Dear FIG,

Response to "An Open Response".

I object strongly to the tone of the above (unsigned) article in Vol. II, No. 6. It is the attitude of the 'computer professional' with access to a large, all singing, all dancing computer looking down his/her nose at the pathetic squirmings of the home computer buff. If this attitude had prevailed, there would be no cheap computers. As it is, a lot of harm is still done by designers making their small computer systems in the image of large computer systems instead of making them like super calculators.

Like many others, I first became interested in FORTH via the August '80 issue of BYTE. One thing that attracted me was the idea that here was a high level language which could be used over the whole range of hardware. There are obvious resemblances between the FORTH and the HP programmable calculator languages and it is reported that FORTH or similar languages are used in hand-held language translators and in one of the hand-held computers. Compare the editorial and, more specifically, Charles Moore's "Characteristics of a FORTH Computer" (p.88) in that BYTE issue with your "Open Response". FORTH is a language in which the user is allowed unparalleled freedom. Please do not insult us by drawing arbitrary limits which will in any case be out of date in a short time.

I will agree that a quart cannot usually be fitted in a pint pot. Solution: devise a means of listing the glossary in such a way that for any word, the indirectly referenced words underlying it can be read. The answer to those wishing to devise minimum systems would then be "go away and get on with it!" Remember that necessity is the mother of invention and the professionals are those who carry on in the wake of the amateurs-- like Einstein--to name but one.

N.E.H. Feilden
47 London Road
Halesworth
Sufolk IP19 8LR
England

P.S. Number typing (e.g., Fixed, floating, double, quad precision, etc.) Surely, all this business of having hundreds of different numbers types is silly, cumbersome, and FORTRAN-like.

Why not forget the whole scheme and do it like BASE. That is to say, have a constant, say NTYPE which tells all operators how many bytes to operate on and whether fixed or floating. It would, of course, be necessary to code all constants and variables in the same way so that when referenced, the appropriate conversions would be done. If this were done in linked lists, then the memory overhead would be very small. The whole thing would be vastly easier to use than what is currently proposed. This suggestion would help to reduce the number of words to remember.

Sounds like you have some interesting and creative approaches to problem solving. You might be interested to know that the author of "Open Response" works on a home size computer. I am sure that no offense was meant and if the author of "Open Response" would like to answer in this space or another column, we will be glad to print it.--ed.

HELP WANTED

Los Angeles Area FORTH PROGRAMMER WANTED -- Contact Linda Stoffer at Pace Personnel, (213) 788-7039.

FORTH, Inc. has the following job openings:

TECHNOTES, BUGS, FIXES

TIPS ON BRINGING UP 8080 Fig-FORTH

Ted Shapin
5110 E. Elsinore Aenue
Orange, CA 92669

Some of the "gotchas" I ran into in bringing up 8080 Fig-FORTH may be helpful to others.

Make sure your assembler will handle lines such as DW A,B-$ correctly. The Boston Systems Office cross-assemblers use the address of the first operand as the value for "$" in the second operand. This leads to a system that will print out the sign-on message but will fail to perform many other operations correctly. I got around this by changing such occurrences to two separate lines: DW A and DW B-$.

The next problem to solve is how to type in the editor screens. It is nearly impossible to type the editor in twice correctly. As R. Allyn Saroyan pointed out, you only need to type in a mini-editor twice. Once, to get it in the dictionary so you can use it, and again, to get it to a screen so you can put in on disk. The mini-editor is simply taken from the implementation model editor screens as follows:

```
HEX : TEXT HERE C/L 1+ BLANKS WORD HERE
PAD C/L 1+ CMOVE ;

:   LINE DUP FFFØ AND 17 ?ERROR SCR @
(LINE) DROP ;

: -MOVE LINE C/L CMOVE UPDATE ;

: P 1 TEXT PAD 1+ SWAP -MOVE ;

DECIMAL
```

Now, proceed to use it to write itself to the disk. You can do this by picking an unused screen, say 85 and typing 85 LIST. Now use "P" to place a line of text on the screen, e.g., 0 P (Mini-editor) will place a comment on line 0 of the current screen.

Type the rest of the lines above and then use the word "FLUSH" to write the mini-editor to disk. Now, when you need to start the system again, just type 85 LOAD and your mini-editor will be put into the dictionary.

Use the mini-editor to type in the Fig-FORTH editor. The string search screen can be omitted if you do not have a version written in highlevel FORTH.

USING ENCLOSE ON 8080

Using ENCLOSE with disk block buffers of 256 bytes each or larger on the 8080 processor.

```
        DB      87H       ;ENCLOSE
        DB      'ENCLOS'
        DB      'E'+80H
        DW      PFIND-9
ENCL    DW      $+2
        POP     D         ;(DE)<-(S1)=DELIMITER CHR
        POP     H         ;(HL)<-(S2)=ADDR. OF TEXT TO SCAN
        PUSH    H         ;(S4)<-ADDR.
        MOV     A,E       ;(a)<-DELIM CHR
        LXI     D,-1      ;INITIALIZE CHR OFFSET COUNTER
        DCX     H         ;(HL)<-ADDR-1
                          ;SKIP OVER LEADING DELIMITER CHRS
ENCL1   INX     H
        INX     D
        CMP     M         ;IF TEXT CHR - DELIM CHR
        JZ      ENCL1     ;THEN LOOP AGAIN
                          ;ELSE NON-DELIM CHR FOUND
        PUSH    D         ;(S3)<-(DE)=OFFSET TO 1ST NON-DELIM
        PUSH    PSW       ;SAVE DELIM CHR ON STACK
        MOV     A,M,      ;IF 1ST NON-DELIM=NULL
        ANA     A
        JNZ     ENCL2
        POP     PSW       ;THEEN DISCARD DELIM CHR
        INX     D         ;(S2)<-OFFSET TO BYTE FOLLOWING NULL
        PUSH    D
        DCX     D         ;(SI)<-OFFSET TO NULL
        PUSH    D
        JMP     NEXT
ENCL2   POP     PSW       ;(A)<-DELIM CHR FROM STACK
        INX     H         ;(HL)<-ADDR NEXT CHR
        INX     D         ;(DE)<-OFFSET TO NEXT CHR
        CMP     M         +IF NEXT CHR<>DELIM CHR
        JZ      ENCL4
        PUSH    PSW       ;SAVE DELIM CHR ON STACK
        MOV     A,M       ;AND IF NEXT CHR<>NULL
        ANA     A
        JNZ     ENCL2     ;THEN CONTINUE SCAN
                          ;ELSE CHR=NULL
        POP     PSW       ;DISCARD DELIM CHR
        PUSH    D         ;(S2)<-OFFSET TO NULL
        PUSH    D         ;(S1)<-OFFSET TO NULL
        JMP     NEXT
                          ;ELSE CHR=DELIM CHR
ENCL4   PUSH    D         ;(S2)<-OFFSET TO BYTE FOLLOWING TEXT
        INX     D         ;(S1)<-OFFSET TO 2BYTES AFTER END OF WORD
        PUSH    D
        JMP     NEXT
```

NOTE: see Mr. Miles' letter in Letters section.--ed.

Mr. William D. Miles
P. O. Box 225
Red Cliffs
Victoria 3496
Australia

CORRECTIONS TO METAFORTH

John J. Cassady
339 15th Street
Oakland, CA 94612

The following corrections to the Fig-FORTH cross-compiler, METAFORTH, by John Cassady should be noted:

page 26 screen 66 line 7 should read

KISR H LXI SRA5 SHLD 12 ORG + LHLD SPHL NEXT JMP

page 38 dumped memory location 798C should be 6A

A few lucky purchasers will have noted that they possess those rare copies of METAFORTH in which pages 8 and 9 are swapped.

METAFORTH, by the way, is a cross-compiler for Fig-FORTH. It can be used to regenerate a FORTH system including the nucleus without resort to an external conventional assembler. This is helpful when modifying low level words, generating "stand-alone" applications, converting to FORTH-79 and the like. A special section is devoted to generating headless configurations with the same or different processor.

METAFORTH is available in hardcopy through: MOUNTAIN VIEW PRESS, PO Box 4656, Mountain View, CA 94040 for $30.00. There are plans to have it available on disk and compatible with several of the popular commercial fig-FORTHs from their respective vendors.

CHANGING 8080 fig-FORTH FOR DISK COPYING

Ted Shapin
5110 E. Elsinore Avenue
Orange, CA 92669

The FigFORTH 8080 implementation uses all bytes of all sectors on the single and double density diskette. This means 2002 sectors on a disk for single sector and 4004 sectors for double. This is not a multiple of eight so the last screen on a disk will be split across two disks. By simply changing the equates for SPDRV1 and SPDRV2 to 2000 and 4000, we will have an even number of screens per disk. This allows a screen disk to be copied from disk A to disk B by using the Fig-FORTH COPY word.

NOTE: Ted has the correct method. Any other system setup that could split screens is incorrect.--ed.

FORTH STANDARDS CORNER

Robert L. Smith

There is a need for a channel of communication regarding the standardization of FORTH. A major topic is the clarification of the FORTH-79 Standard. What changes are required or desirable for clarification or extensions to the Standard? Is the FORTH Standards Team the appropriate mechanism for obtaining a "seal of approval" for corrections and changes to the Standards?

Let us first consider a fairly simple topic, the unsigned count specified in the definition of FILL in the 79-Standard. FILL is defined as follows:

```
FILL     addr  n  byte                  234
```

> Fill memory beginning at address with a sequence of n copies of byte. If the quantity n is less than or equal to zero, take no action.

This is a clear and reasonable unambiguous definition. However, at the Rochester FORTH Standards Conference, there was a strong consensus that the byte count n should be an unsigned number. The restriction in the definition seems to be unnecessary; the only thing to be said in its favor is that it might save a programmer from an inadvertent error (and generally FORTH does not try to save programmers from their errors). If the unsigned FILL were to be the fundamental definition, then the signed version would be trivial to implement. The reverse is more difficult. Thus, the unsigned FILL would lead to better "factoring". Furthermore, a common use for FILL is to preset a large portion of memory. The unsigned version is clearly better suited for this task.

Having said that, what should be done? Since the current definition is unambiguous, and since 79-Standard versions of FORTH currently exist (with several more in advanced stages of development), it seems to me that there should be no change to the 79-Standard in this area. The Standard Team has suggested one mechanism for evolutionary changes in FORTH via "Experimental Proposals". An experimental program would, however, involve a new name for the changed function and could not become a permanently accepted change until two revisions of the Standard. That may or may not be acceptable, depending on the frequency of the revisions.

Please send in material, questions, and comments relevant to FORTH Standards. I will try to cover one or two areas with each issue. Possible topics for next time are the words WORD and +LOOP.

CORRECTION

"Systems Guide to fig-FORTH" by Ting is not available through FIG. Orders for this book, revised 1st edition @ $25.00, should be sent to:

MOUNTAIN VIEW PRESS
PO Box 4656
Mountain View, CA 94040

NEW PRODUCTS

SYM-1 FORTH

Saturn Software Limited has implemented Fig-FORTH for the SYM-1 single board computer. Their implementation takes advantage of many of the features and resources of the SYM-1.

SYM-FORTH 1.0 (disk version) requires 16K of ram, serial terminal, and the dual HDE mini disk system. System has been upgraded to the 79-STANDARD and includes a versatile input line editor, fig-style editor, 6502 assembler, and a cassette interface. This product is also supported by a quarterly newsletter with an initial circulation of 100.

Extras included:

Assembler, editor, cassette interface, plus numerous utilities and demos presented through subscription to newsletter.

Machine on which product runs:

SYM-1, 6502 singleboard computer.

Memory requirements: 16K of ram

Manual:

The 74 page manual includes introductory tutorial material, system information, and glossaries for the FORTH, EDITOR, and ASSEMBLER vocabularies. The manual is available separately for $25 which will be credited towards a later purchase.

Form product is shipped in:

Product is distributed on two 5-1/4 inch diskettes, and boots with 79-STANDARD upgrade installed. (Cassette version is also available which can be upgraded to a disk system at any time.)

Product has five active installations of the disk version (79-STANDARD). There are also 50 installations of the cassette version.

Price:

SFD-1 SYM FORTH FOR DUAL HDE MINI DISK SYSTEM $150 U.S., includes shipping, tax, etc.

Vendor support:

Direct personal support by phone, correspondence, and newsletter.

Order turn around time:

Immediate.

For more information, contact:

Jack W. Brown
SATURN SOFTWARE LIMITED
8246 116A Street
Delta, B.C., V4C 5Y9, CANADA
(604) 596-9764

OSI-FORTH 2.0 / FIG-FORTH 1.1

This is a full implementation of the FORTH Interest Group Version 1.1 of FORTH. It runs under OS-65D3.12 (or 3.0, 3.1), on any disk-based Ohio Scientific system, and has access to all DOS commands and resources.

Extras include resident text editor, Assembler, and utility screens for transferring the system to a new disk, initializing library and system disk block storage tracks, copying screens from disk to disk, and reconfiguring the system memory usage.

Machines:

Ohio Scientific C4P MF, C8P DF, C3, C2-8P DF, C1P MF, and C4P DF. While only one drive is needed, dual drives are supported.

Memory Required: 24K

Manual:

Currently 95+ pages--with new OSI-FORTH Letters added as they are produced. Twenty-four pages of discussion of particulars for OSI, utility screens, and operation of the editor (includes sample edit screen). FIG Installation manual included. Listings of utility and other sample screens. Available separately for $9.95, which is credited toward system purchase.

Media Available: Eight-inch or mini disk.

Approximate number shipped: 25

Price:

$79.95 includes shipping. (Florida residents add 4% sales tax.)

Delivery: 30 days.

Support:

OSI-FORTH Letters subscription available for $4 per year. Contains fixes for any new minor bugs that may be found, as well as listings of application screens donated by users, or developed by Technical Products.

For more information, contact:

Daniel B. Caton
TECHNICAL PRODUCTS COMPANY
4151 N.W. 43 St., #507
P. O. Box 12983
Gainesville, FL 32604
(904) 372-8439

NEW PRODUCT

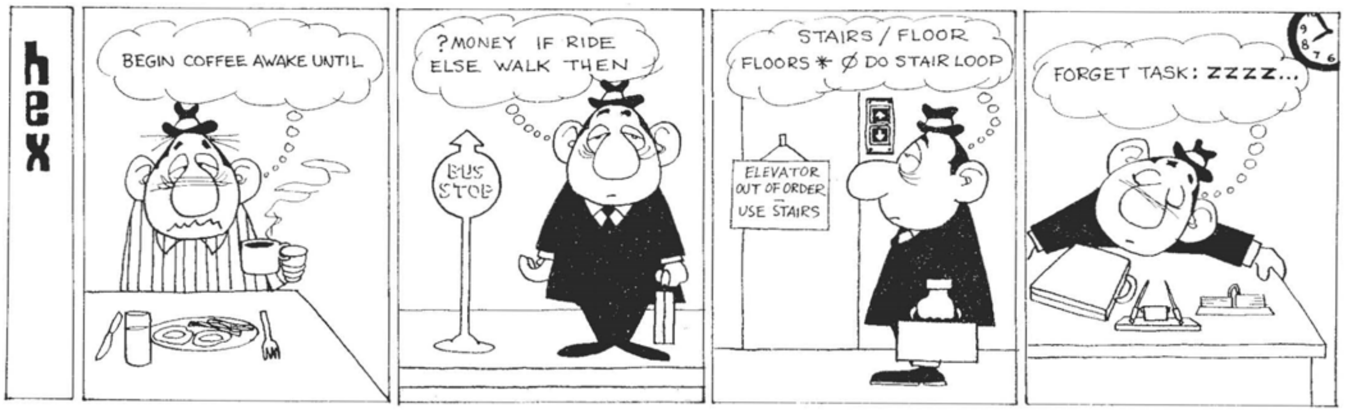
DATRICON FORTH

Datricon now offers D-FORTH, a software package designed for use in conjunction with microprocessor-based, STD Bus compatible products using a Single Board System concept and offering a variety of 68xx/65xx processors. Datricon's single board controllers use interface standards such as the STD Bus, RS232, and RS422 for serial communications and with or without parallel I/O compatible with the popular isolated AC/DC module racks.

For more information, contact:

DATRICON CORPORATION
7911 N.E. 33rd Drive
Portland, OR 97211
(503) 284-8277

Warning--this FORTH is different in names and omitted 'vestigal words'.--ed.


hex
BEGIN COFFEE AWAKE UNTIL
?MONEY IF RIDE ELSE WALK THEN
BUS STOP
STAIRS / FLOOR FLOORS * Ø DO STAIR LOOP
ELEVATOR OUT OF ORDER USE STAIRS
FORGET TASK: ZZZZ...

ELEMENTS OF A FORTH DATA BASE DESIGN

by Glen B. Haydon

In this day and age, data base design and manipulation is one of the major activities best accomplished with computers. In practice, FORTH proves to be an ideal language for developing and using custom data bases. By comparison with other languages, high or low level, FORTH is a winner. It meets the requirements of being interactive and providing documentation as identified by Fred Brooks in his book, THE MYTHICAL MAN-MONTH, as being ideal for the development of new systems. The amazing speed and ease with which custom data bases can be developed, more than justifies the effort required to learn FORTH.

I have developed a number of small data bases of up to 800 records containing 128 bytes each to serve my specific needs. I have also initialized, with simple formatted input and output routines, a custom data base for inventory control in a few hours one evening. Having used languages other than FORTH for similar work, it is highly unlikely I will ever go back to them.

This discussion presents a group of utility FORTH word definitions which facilitate the development of custom data bases and a sample application using these utilities to define a small file. A number of techniques available in FORTH are illustrated.

Some months ago, at a regular monthly meeting of the FORTH INTEREST GROUP in Hayward, the prime mover of the group distributed and discussed several FORTH Screens which provided the foundation for beginning the definition of a data base file. I have modified his Screens slightly and expanded them to provide a general framework with which to define custom accounting data bases. I will assume that the reader has some knowledge of the fig-FORTH Model and proceed with the examination of Screens developed from it. In the discussion, FORTH words are enclosed in single quotes to set them apart from the English words in the text. In FORTH, these words are used without the single quotes.

```
SCR # 21
  ( SCREEN 21: COMMENTS AND LOAD FOR DEMO DATA FILE )
  27 EMIT 69 EMIT CR CR CR CR CR
  ." This demonstration data system provides a pattern for the"
  CR ." further development of any type of data base.  The basic"
  CR ." file formating definitions are on Screens 22 and 23. Some"
  CR ." utilities are on 24, 25, and 26.  The demo file "
  CR ." definitions are on 27. Elementary file manipulation "
  CR ." utilities are on 28.  This model should get you started."
  : PROCEDE CR CR ." ENTER 'Y' TO LOAD SCREENS " KEY 89 = IF
          29 22 DO I LOAD LOOP ENDIF ;
  PROCEDE
  ;S
```

```
SCR # 22
   CR ." SCREEN 22: FILE DEVELOPMENT "
  0 VARIABLE REC#  ( holds the current record number )
  0 VARIABLE OPEN  ( points to current file descriptor )

  : 2@  DUP 2+ @ SWAP @ ;    ( double fetch )

  : LAYOUT  ( leave bytes/record-2, and bytes/block-1 )
    OPEN @ 4 + 2@ ;
  : READ   ( n-th record, on stack, is made current )
    0 MAX DUP OPEN @ 2+ @ < 0=
    IF ." FILE ERROR " QUIT THEN REC# ! ;
  : RECORD   ( leave address of n-th record )
     LAYOUT */MOD OPEN @ @ + BLOCK + ;
  : ADDRESS   ( leave address of the current record )
     REC# @ RECORD ;
  ;S
```

The first two Screens provide eight utility FORTH words for developing a data base file. The comments included in the Screens within parentheses should, combined with the mnemonic nature of the words, give you a clue to what is happening. The first two words are variables used in manipulating the file, 'REC#' and 'OPEN'. '2@' is a FORTH word, and alias for 'D@', which fetches the next two values beginning with the address on the top of the stack and places them on the stack. The word, 'LAYOUT', places two parameters of the new definition of a file on the stack for subsequent use. 'READ' is the first word that one will have occasion to use in routine manipulating records in the data base. It takes the number of the desired record from the top of the stack and, after checking to see that it is a valid record, places its value in the variable 'REC#' which is used to identify the record then under consideration. The word 'RECORD' takes the value for a record number from the stack and returns its address to the stack.

Finally, 'ADDRESS' takes the record number at the variable 'REC#' and using 'RECORD' leaves the address of that record on the top of the stack. With only these eight FORTH words: two variables, one utility word, and four basic words for file referencing, we can proceed to the definition of three defining words in the next Screen.

```
SCR # 23
  CR ." SCREEN 23: FILE DEVELOPMENT - 2 "
: TFIELD  <BUILDS  ( create a text field    )
   OVER , DUP , +  ( leaves file count for this definition )
            DOES>  ( leaves: addr, count  )
  2@ ADDRESS + SWAP ;
: DFIELD  <BUILDS  ( create a data field  )
  OVER , +         ( leaves file count for this definition )
            DOES>  ( leaves address  )
  @ ADDRESS + ;
: FILE             ( create a named storage allocation )
   <BUILDS ,       ( origin block )
     1+ ,          ( number of records in file )
     DUP B/BUF OVER / * ,  ( # bytes per block )
     ,             ( # bytes per record )
    DOES> OPEN ! ; ( when file name used, point to )
;S                 ( its descriptor parameters.    )

MSG # 15
OK
```

The three words on the next Screen are called defining words because they are used to define new FORTH words as the names of fields in our record and to define the name for the file we are defining, each with specific properties. These words utilize the combination of the FORTH primitives '<BUILDS' and 'DOES>' which are present in the Model. It may take some time to fully appreciate what these primitive words accomplish and the way they work. Perhaps an examination of what they are doing in this Screen will help you understand their function.

Two types of record fields are distinguished and defined with separate words, a numerical or data field and a text field. The first word, "DFIELD', is used to add to a record being defined, a field containing the number of bytes given on the top of the stack and gives that field a name. In subsequent use, that newly defined word (data field name) will cause the address of that field in the record whose value is currently in the variable 'REC#' to be left on the stack. This word is used to identify the location in a record where a numerical value is to be stored in a binary form. I call it a "data field", in contrast with a "text field" in which the length of the field should also be immediately available. Thus 'TFIELD' is used to define a "text field" which will identify a field in the new record with a length in bytes given on the top of the stack and gives that field a name. In subsequent use, that newly defined word (text field name) will cause not only the address of that text field in the record whose value is currently in the variable 'REC#' to be left on the stack, but also the length of that field. The length is convenient when the primitive word 'TYPE' is used to print the character string in that field. Obviously the length is not needed in a data field. Thus, provisions are made for defining two types of fields in a record. As new fields are added to a record in the course of its definition, the current length of the record is maintained on the top of the stack.

```
SCR # 24
  CR ." SCREEN 24: DOUBLE PRECISION ARITHMETIC "
: D/      ( d, u --- d    )
     SWAP OVER /MOD >R SWAP U/ SWAP DROP R> ;
: D*      ( d, u --- d    ) .
     DUP ROT * ROT ROT U* ROT + ;
: D/MOD   ( d, u --- r, q )      U/ ;
: D@      ( addr --- d    )      DUP 2 + @ SWAP @ ;
: D!      ( d, addr ---   )     DUP ROT SWAP ! 2 + ! ;
: DDUP    ( d --- d, d    )     2DUP ;
: DSWAP   ( d1, d2, --- d2, d1 )   ROT >R ROT R> ;
: DDROP   ( d1, d2 ---    )      DROP DROP ;
;S
```

Once the definition of the fields in a record is completed, the value of the record length remains on the stack. To this we need to add values for the number of records we wish to include and finally, the block number in which the records are to start, before we can use the defining word 'FILE' to give the file a name. Later when the new file name is used, the address of the necessary file parameters is placed in the recently defined variable 'OPEN' as required for access to any given record with the words defined in the first Screen.

With these two Screens, we have the file utilities necessary to define a new file. However, several characteristics of the particular implementation of FORTH which is being used are important. Most systems created under the Model have 128 bytes per block although any multiple of 128 can be used. In these sytems then, the largest record length can be no longer than 128 bytes, but with a larger block size, larger records can be used. In order to take maximum advantage of the block size, it should be very nearly equal to a multiple of the record length. For example, a record length of 70 bytes would not leave enough room in a 128 byte block for a second record and in this case, 58 bytes of space would be wasted. If need be, such a designed file would work, but at the expense of memory space. Also, the initial block to be used in the definition created by the word 'FILE' must be chosen according to the block size for the particular implementation. For example, block 400 in an implementation with 128 bytes per block would be block 50 in an implementation with 1024 bytes per block. Although, I find a block size of 1024 to be more efficient and use it routinely, the Screens presented here have been written for and tested on an implementation with 128 bytes per block.

Before starting with a discussion of an example of the application of these file development utilities, several Screens of utilities for use in the input and output of numerical data will prove to be most helpful. These include a group of double precision utilities, date compression and expansion routines, a numerical routines for handling dollar amounts and storing them as double precision integers.

```
SCR # 25
  Ø  CR ." SCREEN 25: DATE COMPRESSION AND EXPANSION "
  1 : DATEBIT ( converts date input to 2 bytes  )
  2        32 D* D+ 13 D* D+ DROP   ;
  3 : ?DATE     ."  ( MM/DD/YY ) "
  4        QUERY 47 WORD HERE NUMBER 47 WORD HERE NUMBER BL WORD
  5        HERE NUMBER DATEBIT ;
  6 : .DATE       ( formats 2 byte date code and prints )
  7        Ø 13 U/ 32 /MOD ROT 100 * ROT + Ø 100 D* ROT Ø D+
  8         <# # # 47 HOLD # # 47 HOLD # # #> TYPE ;
  9     ;S
 1Ø
 11
 12
 13
 14
 15
```

The double precision integer utilities are used in date compression and expansion as well as in the double precision integer operations for dollar amounts. These are simple extensions from the limited double precision words found in the Model and should require no further explanation. The input on the stack before executing the word and the output left on the stack afterwards are indicated in the format used in the fig-FORTH GLOSSARY. You will note that several of these are mixed double and single precision operations which are sufficient for the requirements of this program.

The date compression routine is really simple. When I find the time I will develop an algorithm to convert the date to a true Julian day and store the least significant value. This would make calculation of the time between two given dates easy. In the meantime, the present routine allows one to enter the date as numerical values separated by slashes, a commonly used format, and reduce the value to a single 16 bit integer requiring only two bytes for storage. The routine provides an example of using a delimiter other than a space to parse 'WORD' and the use of 'NUMBER' to interpret a numerical value without searching the dictionary. After the parsing of the input, three double precision numbers are left on the stack. The word 'DATEBIT' defines a simple algorithm which is applied to reduce these three double precision values to 16 bits. The execution of '?DATE' first prompts with the format to be used, then waits for the value to be entered. The value is then converted to the 16 bits and left on the stack for starage. since '.' is used to conote "print" in FORTH, '.DATE' is defined to print a properly formatted date from a 16 bit integer on the stack. This routine is useful as an example of conversion of a binary value to a text string for printing.

```
SCR # 26
  Ø  CR ." SCREEN 26: ?SAMOUNT AND .SAMOUNT  "
  1  ( define action for each scale case  )
  2 : ØSCALE 100 D* ;   : 1SCALE 10 D* ;   : 2SCALE ;
  3 : 3SCALE ." INPUT ERROR " CR ;
  4  ( define scale case and extend for each with 'CFA' )
  5 ' ØSCALE CFA VARIABLE NSCALE ' 1SCALE CFA , ' 2SCALE CFA ,
  6   ' 3SCALE CFA ,
  7  ( scale double precision value according to 'DPL' )
  8 : SCALE DPL @ 3 MIN 2 * NSCALE + @ EXECUTE ;
  9  ( wait for decimal value and scale it - leave value on stack )
 1Ø : ?SAMOUNT QUERY BL WORD HERE NUMBER SCALE ;
 11  ( print d from stack as $ and right justify in 8 spaces )
 12 : .SAMOUNT
 13   DUP ROT ROT DABS <# # # 46 HOLD #S SIGN #>
 14   36 EMIT DUP 8 SWAP - SPACES TYPE ;
 15 ;S

MSG # 15
OK
```

Finally, we have a Screen to define some FORTH words used to input and output dollar amounts and convert them to and from double precision 32 bit integers with the necessary scaling for the location of the decimal point. In FORTH, the use of a decimal point forces an input number to a double precision integer which takes four bytes. A convenient FORTH primitive word, 'DPL' for decimal point locator, keeps a count on the number of digits entered following the decimal point. Utilizing this value as an input for a case type word, the numerical value entered can be scaled properly, regardless of how many digits are entered to the right of the decimal if any. This method of executing a case like routine is straight forward. First, the action to be taken in each case is defined. '0SCALE' means that there were no digits to the right of the decimal which requires that the entered double precision integer must be multiplied by 100. In a similar manner '1SCALE' is used meaning that there was only one digit entered following the decimal point and the entered double precision integer must be multiplied by 10. '2SCALE' does nothing since no scaling is needed. Finally, if more than 2 digits are entered an error must have been made an an appropriate error message is given. Once each of the cases is defined, their code field addresses, 'CFA', can be stored beginning with the address of a defined variable 'NSCALE' and extending into the alloted space. The word 'SCALE' then finds the value of the variable 'DPL' and counts over to the proper code field address which is then placed on the stack and the selected word is executed.

After this scaling operation, the word to input a dollar amount '?$AMOUNT' is defined which leaves the scaled double precision integer on the stack ready to be stored. Finally, a routine defined by the word '.AMOUNT' connoting "print dollar amount" will print the double precision integer on the top of the stack as a dollar value right justified in eight spaces.

There are certainly other and probably better ways to accomplish the work done by these three Screens of utilities, but they work. The way they work provides some examples of the beauty of FORTH as it exists in the Model.

With these five Screens, we can very quickly define a record for a data base with custom selected fields and then the associated file characteristics. In the past, I have several times included in a data base values calculated from other values in the base. On occasion, it has been necessary to change one of the original values. This has always required that the calculated fields be redone, too. I now find that it is more convenient to enter only the basic data. All calculations can be made while the output is being formatted and printed with no significant loss of time. The slowest part of printing the formatted result is the delay in the output device.

```
SCR # 27
 CR ." SCREEN 27: DEMO FILE - RECORD GENERATION "
 0  2  DFIELD  TAG     ( a tag  )
   30  TFIELD  NAME    ( item name  )
    2  DFIELD  DAY     ( the date  )
    4  DFIELD  DOLLAR ( a dollar amount )
      200 ( number of records ) 400 ( starting block )
      FILE  DEMO
 : !NAME  ( wait for name then store it in record )
      NAME DROP 30 32 FILL QUERY 1 TEXT PAD COUNT
      NAME ROT MIN CMOVE UPDATE ;
 : .NAME  ( print name field )   NAME TYPE ;
          ( the rest follow in the same way  )
 : !DAY ?DATE DAY ! UPDATE ;  : .DAY  DAY @ .DATE ;
 : !DOLLAR ?$AMOUNT DOLLAR D! UPDATE ;
 : .DOLLAR DOLLAR D@ .$AMOUNT ;
 : .REC CR REC# @ 3 .R 2 SPACES .NAME .DAY 2 SPACES .DOLLAR ;
```

As an example of the definition of a new data base, I have chosen one in which each record would be allotted 4 fields for a two byte tag, a 30 byte stock name, a two byte date, and a 4 byte stock price. Though little could be done with this as a data base, it does provide an example of each type of input. Finally, a simple set of routines is given to clear the records, input new records, and print out a list of the records in the file.

As a matter of convention, I give each field a name with no prefix. Thus, a data field name will leave an address on the stack and a text field name will leave an address and count on the stack. By using the FORTH connotations of'!' for store and '.' for print, I define some utilities for inputting data and text and printing out the respective fields. From these utilities, I can assemble an input format and an output format as desired. I have not included routines for error checking which would be most desirable especially in a hostile environment.

Now, to examine the actual example of the definition of a file which we will call 'DEMO'. Each record will begin with zero offset from the record address and a '0' is entered followed by '2' for a two byte length of a data field to be named 'TAG'. Many occasions in later manipulation of records make it desirable to have such a field for adding flags, etc. Following this definition, the length value of 2 is left on the stack so that for the next field, only its length need be entered. In this case, a text field of 30 bytes which is given the name 'NAME' which then leaves the value of 32 (the length of the 'TAG' field plus the 'NAME' field) on the stack. Then a two byte data field, 'DAY' is reserved for a 16 bit compressed date and then a four byte data field 'DOLLAR', for a double precision integer value of a dollar amount. With this, the 4 fields within the record of a new file are defined. Next, we will define the file name. According to the utility for generating a new file, we must first add to the value of the record length remaining on the stack, a value for the number of records we plan to include in the file and then the first block number to be used as determined by the FORTH implementation in use. Then, we use the word 'FILE' to create a file with these paramters and give it the name 'DEMO'. The data base file is now defined. For the record number whose value is in the variable 'REC#', we can place the value of the address of the data fields and the address and count of the text fields on the stack by simply entering the field name. Next, a few simple utilities will make accessing these new fields easier.

Remembering the connotations associated with the FORTH words '!' and '.' we will define words to input data or text to the appropriate fields of that record whose value is currently in the variable 'REC#'. These are simple file primitives which will then be available for routines to format input and outputs as desired. The field 'TAG' is not used at this time and specific routines are not defined. To store a name in the name field, we define the word '!NAME'. This routine first fills the existing field with blanks, ASCII 32 (decimal) and then pauses for input from the keyboard. The input text is truncated to the maximum length of the text field if necesary and then moved to that field. In order to output the name in the field, we define the word '.NAME'. In a similar manner, we define '!DAY' to store a 16 bit integer value of a date which has been compressed into that field. In the earlier utilities, we have already defined '?DATE' which waits for a date to be input and leaves the compressed value on the stack. All that is necessary is to put the address of the field on the stack with 'DAY' and then store the encoded date there. We then define '.DAY' to output the date stored in the 'DAY' field. We get the 16 bit value stored there to the top of the stack and use the previously defined word '.DATE' to output it in the proper format. Finally, we define '!DOLLAR' to parse a dollar value input with a decimal point in any location and scaled to a double precision number which is then stored in the proper field. In a similar manner'.DOLLAR'is defined to format the stored double precision integer to a right justified eight digit number preceded by a dollar sign. With these definitions, we have completed a set of FORTH words to input and output data from records in our data base.

Immediately after putting data into a record, it is often desirable to see what is actually present in that record. The values in each byte of a record can be displayed using a dump routine. Simply place the desired record address on the top of the stack by entering the record number followed by our file utility word 'READ' and 'ADDRESS' followed by the length of the record and the word for your dump routine. But the byte values printed out in hex or decimal are not really all that helpful. It is hard to interpret the numerical value in their byte pattern. A convenient word '.REC' is defined to print out the current record number followed by the formatted output of the value in each field using the above utilities and an appropriate number of spaces and carriage returns. This is the most rudimentary form of a formatted output. If desired, the output could be presented in reverse vidēo by a slight modification of this routine. It could also be placed anywhere on the screen.

```
SCR # 28
CR ." SCREEN 28: DEMO FILE - CLEAR.DATA, INPUT, OUTPUT "
 ( clear especially tag in the Ø record in file )
: CLEAR.DATA Ø READ TAG 128 Ø FILL UPDATE ;
 ( example of formatting for input  )
: INPUT Ø READ TAG @ 1+ UPDATE DUP TAG ! READ
   CR CR ." ENTER NAME                          --> " !NAME
      CR ." ENTER DATE      --> " !DAY ( has a format prompt )
      CR ." ENTER AMOUNT                        --> " !DOLLAR
   .REC FLUSH ;       ( save this record on disk )
 ( list files 1 through the number in TAG of Ø record )
: OUTPUT Ø READ TAG @ DUP Ø= IF CR CR ." EMPTY FILE "
      DROP ELSE 1+ 1 DO FORTH I READ .REC LOOP ENDIF CR CR ;
;S
```

Finally, a few examples of formatting input and output routines are shown on the last Screen. First, it is desirable to clear all data in a file with a word 'CLEAR.DATA' before entering new data. This particular definition clears only the first block, all that is necessary in this application. You should be able to modify the definition of this word to meet the requirements of your application and particular implementation of FORTH.

I use the 0 record in a file for a variety of information about the file which I can address directly from the address of its first byte without using the field definitions or I can use specific bytes or fields in ways other than I have defined them. In this example, I use the value in the integer at the field 'TAG' in the 0 record to keep track of the last record currently in the file. When this record is cleared with 'CLEAR.DATA', a value of 0 is present in the location of 'TAG' which means that there are no records present. '0 READ' places the value of 0 in the variable 'REC#' and then 'TAG' places the address on the top of the stack and '@' gets that value, the last record number used in the file. To add a new record, this value is incremented and then duplicated on the stack. The top copy is stored back in the field of 'TAG' in the 0 record which is updated. Then the second copy is placed in the variable 'REC#' and we are ready to fill in the information for the next record.

A series of prompts can be formatted on the screen in any convenient arrangement as in this example. Following the desired prompt for each field, the previously defined word is used to get the information for the field and store it there. After entering a record, it is always nice to see the data you actually put in. This is done with the word '.REC' followed by the FORTH primitive 'UPDATE' to flag the buffer as altered and 'FLUSH' to save the new record on the disk in the file. This assures that the image of the record which is displayed is the version saved on the disk.

An output format can be developed in a similar manner. In this example I have included a check to see if there are any records in a file because the 'DO'... 'LOOP' will always print one loop and peculiar output is generated if the bytes in the fields are all set to zero. This output routine presents a simple list of the record numbers and the formatted content of the fields.

In conclusion, I find this approach to file definition is time saving and hope that you will find it useful. The discussion of the FORTH utilities used to define a new data base file and the example example of handling data, provides some elaboration of the information included on the Screens. This will be a review for one who already has learned the primitives in the FORTH Model and understands how the language works, but perhaps the discussion of these Screens will help those less experienced. There is nothing sacred about the techniques used here. Modify the various words to suit your particular needs. It is easy enough to develop new formats interactively. However, I would encourage you to utilize and build on the standards of the fig-FORTH Model. When the '79 Standards become generally available, it should be relatively easy to update your Screens without changing the format of the record file.

The importance of utilizing an accepted standard in developing programs for ultimate use in a wide variety of implementations of FORTH cannot be overemphasized.

I wish to thank Bill Ragsdale for his encouragement to write this discussion based on his presentation to the FORTH INTEREST GROUP at one of their monthly meetings last year.

APPLICATION NOTE:

These FORTH routines have been developed on a FORTH OPERATING SYSTEM for the HEATHKIT H89. This system is available from the MOUNTAIN VIEW PRESS, Box 4656, Mountain View, CA 94040. The compiled FORTH program image can be saved on disk and will be up and running in less than four seconds from a cold boot. The system has 1024 byte blocks which also increases the speed of operation.

However, after develoment, the Screens were loaded on a FORTH implementation derived from the fig-FORTH FOR 8080 ASSEMBLY SOURCE LISTING which is available from the FORTH INTEREST GROUP, Box 1105, San Carlos CA 94070, in printed form and already on disk also from the MOUTAIN VIEW PRESS. This version has 128 byte block and operates in conjunction with CP/M. To this has been added the fig-EDITOR from the fig-FORTH INSTALLATION MANUAL and a single extension, DUMP, used to illustrate the appearance of the records as stored in a block.

The printed session illustrated was made using the CP/M control P to echo the output on the printer. The session starts with CP/M loaded and its usual prompt. The CP/M file, FORTH60.COM, is the object module of the fig-FORTH Model. The warning messages are not on Screens 4 and 5 and the warning flag is turned off. Then, the Screens for the fig-EDITOR and a good dump routine are loaded. Finally, the Screens discussed are loaded. The file 'DEMO' is called and the application of some of the file utilities is illustrated. This presentation will hopefully assure that there are no errors in the printed Screens.

BIBLIOGRAPHY

Brooks, F. P., Jr., THE MYTHICAL MAN-MONTH, Addison-Wesley Publishing Company, 1975.

fig-FORTH INSTALLATION MANUAL, GLOSSARY, MODEL, Forth Interest Group, Box 1105, San Carlos, CA 94070.

fig-FORTH FOR 8080 ASSEMBLY SOURCE LISTING, Forth Interest Group, Box 1105, San Carlos, CA 94070.

```
A>
A>FORTH60       ( fig-FORTH Model )


8080 fig-FORTH 1.1
 OK
 OK
0 WARNING !     ( Warning messages not on Screens 4 & 5 )
 OK
 OK
 OK
47 LOAD         ( fig-EDITOR )
 R MSG # 4  I MSG # 4
FLUSH MSG # 4  OK
 OK
 OK
 OK
49 LOAD         ( My version of a good dump )


SCREEN 49: GOOD DUMP
OK
 OK
 OK
21 LOAD         ( Loads Screens discussed )



This demonstration data system provides a pattern for the
further development of any type of data base.  The basic
file formating definitions are on Screens 22 and 23. Some
utilities are on 24, 25, and 26.  The demo file
definitions are on 27. Elementary file manipulation
utilities are on 28.  This model should get you started.

ENTER 'Y' TO LOAD SCREENS
SCREEN 22: FILE DEVELOPMENT 2@ MSG # 4
SCREEN 23: FILE DEVELOPMENT - 2
SCREEN 24: DOUBLE PRECISION ARITHMETIC
SCREEN 25: DATE COMPRESSION AND EXPANSION
SCREEN 26: ?SAMOUNT AND .SAMOUNT
SCREEN 27: DEMO FILE - RECORD GENERATION
SCREEN 28: DEMO FILE - CLEAR.DATA, INPUT, OUTPUT OK
 OK
 OK
 OK

DEMO OK
CLEAR.DATA OK
OUTPUT

EMPTY FILE

OK
INPUT

ENTER NAME                       --> ZENITH
ENTER DATE      -->  ( MM/DD/YY ) 4/21/81
ENTER AMOUNT                     --> 18.50
  1  ZENITH                          04/21/81  $   18.50OK
 OK
INPUT

ENTER NAME                       --> IBM
ENTER DATE      -->  ( MM/DD/YY ) 4/21/81
ENTER AMOUNT                     --> 60.
  2  IBM                             04/21/81  $   60.00OK
 OK
INPUT
```

```
ENTER NAME                      --> DEC
ENTER DATE      -->  ( MM/DD/YY ) 4/21/81
ENTER AMOUNT                    --> 103.5
  3  DEC                            04/21/81  $  103.50OK
 OK
 OK
0 READ ADDRESS HEX 80 DUMP DECIMAL
  58CA     3  0  0  0  0  0  0  0   0  0  0  0  0  0  0  0   ................
  58DA     0  0  0  0  0  0  0  0   0  0  0  0  0  0  0  0   ................
  58EA     0  0  0  0  0  0  0  0  5A 45 4E 49 54 48 20 20   ........ZENITH
  58FA    20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20
  590A    20 20 20 20 20 20 B5 84   0  0 3A  7  0  0 49 42         ....:...IB
  591A    4D 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20   M
  592A    20 20 20 20 20 20 20 20  20 20 20 20 B5 84  0  0               ....
  593A    70 17  0  0  0  0  0  0   0  0  0  0  2  0  0  0   p...............
OK
 OK
 OK
OUTPUT
  1  ZENITH                         04/21/81  $   18.50
  2  IBM                            04/21/81  $   60.00
  3  DEC                            04/21/81  $  103.50

 OK
 OK

                                        OK
 OK
 OK
: STATEMENT CR CR 20 SPACES ." STATEMENT " CR CR OUTPUT
      CR CR ." TOTAL VALUE "  33 SPACES  0 0 0 READ TAG @ 1+ 1
      DO I READ DOLLAR D@ D+ LOOP .SAMOUNT CR CR CR ; OK
 OK
 OK
STATEMENT

                    STATEMENT


  1  ZENITH                         04/21/81  $   18.50
  2  IBM                            04/21/81  $   60.00
  3  DEC                            04/21/81  $  103.50



TOTAL VALUE                                   $  182.00


OK
```

fig-TREE TELECONFERENCE

(415) 538-3580

If you are an active FORTH programmer, or just have an interest in FORTH, you will want to save this phone number. With your terminal or computer and a modem, the number will get you on-line to a dynamic data-base on FORTH.

Want to ask a question? Want to know where and when the next important FORTH Interest Group seminar, meeting, workshop, or other event is going to be? The fig-Tree has a calendar section where you can find out about these events and let others know about yours. Want to find out about FORTH-related software, products and services?

Dial-up the fig-Tree for on-line information. Use any 300 or 110 baud modem, and type several carriage returns; then the system is self-instructing.

HELP WANTED

FORTH PROGRAMMERS Experienced with mini/micro computers and peripherals to produce new polyFORTH systems and scientific/industrial applications. Degree in science or engineering and knowledge of FORTH essential.

PRODUCT SUPPORT PROGRAMMER Responsible for maintaining existing list of software products, including the polyFORTH Operating System and Programming Language, file management options, math options and utilities and their documentation. Also provide technical support to customers.

PROJECT MANAGER Supervise applications and special systems programming projects: writing proposals, setting technical specifications, customer liaison, hands-on programming, and supervision of senior programmers. Extensive FORTH programming experience, some scientific or engineering background and management skills required. Bachelors degree or equivalent.

Contact: Min Moore
FORTH, Inc.
2309 Pacific Coast Hwy.
Hermosa Beach, CA 92054
(213) 372-8493

WRITERS WANTED
ANY FORTH SUBJECT
SEND TO:

FORTH INTEREST GROUP
P.O. Box 1105
San Carlos, CA 94070

INCREASING fig-FORTH DISK ACCESS SPEED

by Michael Burton

Anyone who has used CP/M and has then used 8080 fig-FORTH will have noticed that CP/M is much faster than fig-FORTH when reading or writing data on floppy diskettes. The reason for this apparent speed difference lies in the manner in which CP/M stores its files as opposed to how fig-FORTH stores its screens. (Editor's note: Speed is also reflected in hardware details such as interleaved formatting and direct memory access. It is not necessarily a FORTH characteristic.) I shall attempt to explain the difference.

A single-sided 8" diskette formatted in the normal manner contains 77 tracks, with each track containing 26 sectors with 128 bytes of data in each sector. In order for the disk controller to be able to find a particular sector in a given track, header data is stored on the diskette just prior to each 128 byte data block - a sort of preamble. Among other information in this preamble is the sector number. A format program writes this information on each track in a consecutive manner; in other words, immediately following the index hole pulse is sector 1,2,3, ... 26.

A program that reads a sector must first select the proper track and proper sector, then must read that sector's data and store it someplace for use. It is fairly easy to select the proper track and sector and read the data; the problem comes in trying to read two consecutive sectors. There is not enough time between the time when the first sector's data is read and the time when the next sector is available, to store the data from the first sector and request the data from the second sector. This means that reading consecutive sectors 5 and 6, for example, requires a minimum of two revolutions of the diskette.

CP/M accesses files faster than fig-FORTH accesses screens because the files are not stored in consecutive sectors. CP/M uses a translation table to tell it which sector to use. Someone figured out that while storing the data from one sector, about five more sectors go by before CP/M is able to read another sector. So instead of storing a file in sectors 1,2,3 ... it uses its translation table and stores the file in sectors 1, 7, 13, etc. This means that 1024 bytes of information can be read or written in two or three revolutions of the diskette instead of eight.

What can be done about the manner in which fig-FORTH reads/writes screens? A CP/M-style translation table could be added to fig-FORTH, but that would make the diskettes, and the FORTH program, incompatible with the rest of the FORTH world. Instead, the diskettes can be formatted to look like a CP/M translation table, which is extremely easy and still allows compatibility. A diskette would look like this:

Sector

Old format:	1	2	3	4	5	6	7	8	9	10	11	12	13
New format:	1	14	10	23	6	19	2	15	11	24	7	20	3
Old format:	14	15	16	17	18	19	20	21	22	23	24	25	26
New format:	16	12	25	8	21	4	17	13	26	9	22	5	18

Most format programs use an incrementing register to supply the proper sector number when formatting. To implement the translation scheme, a table must be added to the program and must be accessed in place of the sector register when formatting the diskette.

With this new format, fig-FORTH still reads 'consecutive' sectors (1, 2, 3, etc.), but they are available sooner. Using the new style format, fig-FORTH should be able to read or write a screen in two or three disk revolutions instead of eight.

Two simple tests were run to determine how this affects fig-FORTH performance:

	Old Format	New Format	Speed Increase
LOAD 13 Screens:	1:28	1:17	12.5%
COPY 50 Screens:	2:24	:53	63.2%

If you can't reformat your diskettes, you may choose to copy diskettes simulating interleaveing by this program. The multiple screen copy definition used for the second test is listed in screen 167.

SCR # 167

```
( BATCH-COPY FROM #, TO #, DEST # )

: BATCH-COPY ( FROM-SCR, TO-SCR, DEST-SCR --- )
  EDITOR
  ROT ROT  ( DEST, FROM, TO                  )
  1+  SWAP ( DEST, TO+1, FROM                )
  DO
     I     ( DEST, I                         )
     OVER  ( DEST, I, DEST                   )
     COPY  ( from screen I to DEST           )
     1+    ( DEST+n                          )
   LOOP
   DROP FORTH ;



ok
```

```
SCR # 51
 ( Music - Experimental Constructs                         Ø52481-MPB )
 ( Music with Digital Research: Computers S-100                       )
 (  Sound Effects Board - Uses two AY3-891Ø                           )
 (  sound generator ICs                                               )

 ( 1/O port and variable declaration                                  )

 HEX
     ØE8 CONSTANT REG-PORT   ØE9 CONSTANT DAT-PORT
     ØØØ VARIABLE VOICE      ØØØ VARIABLE CHANNEL
     ØFF VARIABLE EMASK1     ØFF VARIABLE EMASK2
     ØØ8 VARIABLE ALV1       ØØ6 VARIABLE ALV2-6
    1F4Ø VARIABLE DVAL       ØFF VARIABLE IVAL
 -->


SCR # 52
 ( Music - Experimental Constructs                         Ø52481-MPB )

 ( Music board access definitions                                     )
 : REG#  REG-PORT  CHANNEL  @  +  P!  ;                 ( REG# --- )
 : DATA  DAT-PORT  CHANNEL  @  +  P!  ;            ( REG-DATA --- )

 ( Tone enable definitions      )
 : ENABLE1 7 REG-PORT P! DAT-PORT P! ;                ( EMASK1 --- )
 : ENABLE2 7 REG-PORT 2 + P! DAT-PORT 2 + P! ;        ( EMASK2 --- )

 ( Set music board registers definitions                              )
 : SET  ?TERMINAL IF ØFF ENABLE1 ØFF ENABLE2 ØFF EMASK1 !
    ØFF EMASK2 ! ABORT THEN REG#  DATA  ;   ( REG# REG-DATA --- )
 : TON-PER  Ø  VOICE  @  2  *  +  SET
            1  VOICE  @  2  *  +  SET  ;       ( COARSE FINE --- )
 : AMP      8  VOICE  @  +  SET  ;    -->        ( AMPLITUDE --- )
SCR # 53
 ( Music - Experimental Constructs                         Ø52481-MPB )

 ( Select voice and set enable mask                                   )
 : sv1 <BUILDS C, C, C, DOES> DUP 1+ DUP 1+ C@ EMASK1 @ AND
                              EMASK1 ! C@ VOICE ! C@ CHANNEL ! ;
   ØFB 2 Ø sv1 v1    ØFD 1 Ø sv1 v2    ØFE Ø Ø sv1 v3
 : sv2 <BUILDS C, C, C, DOES> DUP 1+ DUP 1+ C@ EMASK2 @ AND
                              EMASK2 ! C@ VOICE ! C@ CHANNEL ! ;
   ØFB 2 2 sv2 v4    ØFD 1 2 sv2 v5    ØFE Ø 2 sv2 v6

 ( Reset voice bit in enable mask                                     )
 : vd1 <BUILDS C, DOES> C@ EMASK1 @ OR EMASK1 ! ;
   4 vd1 v1dis    2 vd1 v2dis    1 vd1 v3dis
 : vd2 <BUILDS C, DOES> C@ EMASK2 @ OR EMASK2 ! ;
   4 vd2 v4dis    2 vd2 v5dis    1 vd2 v6dis      -->

```

MUSIC GENERATION IN FORTH

by Michael Burton

The General Instruments programmable sound generator (PSG), the AY3-8910, can be used to produce very acceptable three voice music when properly programmed. FORTH's background as a device control language makes it a good choice to use with the PSG for music production.

The programmable sound generator is capable of producing sound on three separate analog channels. The amplitude and/or envelope of each of these channels is also separately controllable. Although the PSG is used by several manufacturers on their music boards, the board that was used for the development of the attached music constructs is the S-100 Sound Effects Board produced by Digital Research Computers of Garland, Texas. This particular board contains two AY3-8910 chips, allowing up to six voices to be generated simultaneously.

Now, for an explanation of the music screens. Screen 51 consists of definitions of I/O port values and variable declarations. The variable ALV1 is the melody voice amplitude (voice one) and the variable ALV2-6 is the harmony voices amplitude (voices two through six). These amplitudes may be varied from 0 (off) through 15. It is a good idea to keep the harmony amplitude about two steps lower than the melody amplitude, in order to make the melody stand out. The variable DVAL controls the length of the notes, DVAL being the length of a whole note. The variable IVAL controls the length of the slight no-tone period between notes. Together, DVAL and IVAL control the song's tempo. Experimentation is necessary with these two variables to produce the proper tempo for a particular song.

Screen 52 contains all the definitions necessary to access the S-100 Sound Effects Board in order to play music. The only PSG registers currently being used in music generation are the tone period, enable and amplitude registers. Note the use of the 8080 fig-FORTH peculiar word

P!. P! simply sends a data byte to a particular I/O port.

Screen 53 marks the start of the actual definitions used in producing a song. The definitions v1 through v6 are used to select voices 1 (melody voice) through 6. These definitions do not turn the appropriate voice on, they merely select it so that a tone period (note) may be set for that voice. Voices are actually played only when a note duration is selected. The definitions v1dis through v6dis are used to disable a particular voice the next time a note duration is executed. They do not turn a voice off if it is currently being played, they just turn it off the next time it is supposed to be played.

Screens 54 and 55 define the musical notes. The lowest note that may be played is b of octave 0, and the highest note that may be played is b of octave 8.

Screen 56 contains the definitions RINIT, VON and VOFF. RINIT initializes all the registers on each PSG. RINIT is the only place where the amplitude of the voices is initialized, and should be used before playing any music. The definitions VON and VOFF are used to turn all selected voices on and off. They are used inside the note duration definitions and are not meant to be used in a song definition.

Screen 57 contains definitions for rest durations, from a sixty-fourth rest (fr) to a whole rest (wr). It also contains definitions for slur note durations, from a sixty-fourth slur (fs) to a whole slur (ws). A slur note is one that does not go off after its duration is finished, allowing a smooth transition between notes when desired. Screen 57 also contains definitions for dotted slur durations, from a sixty-fourth dotted slur (fds) to a whole dotted slur (wds).

Screen 58 is the last of the music constructs screens, and contains definitions for note durations, from a sixty-fourth note (f) to a whole note (w). Note that after the note is turned off, a slight delay (IVAL) is introduced so that the notes will be distinct from one another. Screen 58 also contains definitions for dotted notes, from a dotted sixty-fourth (fd) to a dotted whole note (wd).

There is room for improvement in these music definitions. Control of the notes' envelope could be introduced to simulate other musical instruments, and restrictions imposed by the non-interrupt nature of the note duration generation could be eliminated. These exercises will be left to other aspiring composer/ programmers.

```
SCR # 54
  ( Music - Experimental Constructs                       Ø52681-MPB )

  ( Note definitions                                                  )
  : n <BUILDS C, C, DOES> DUP 1+ C@ SWAP C@ TON-PER ;
    ØF ØD2 n bØ     ØE ØEE n c1     ØE Ø18 n c#1    ØD Ø4D n d1
    ØC ØEE n d#1    ØB ØDA n e1     ØB Ø2F n f1     ØA ØØF n f#1
    Ø9 ØF7 n g1     Ø9 Ø68 n g#1    Ø8 ØE1 n a1     Ø8 Ø61 n a#1
    Ø7 ØE9 n b1     Ø7 Ø77 n c2     Ø7 ØØC n c#2    Ø6 ØA7 n d2
    Ø6 Ø47 n d#2    Ø5 ØED n e2     Ø5 Ø98 n f2     Ø5 Ø47 n f#2
    Ø4 ØFC n g2     Ø4 ØB4 n g#2    Ø4 Ø70 n a2     Ø4 Ø31 n a#2
    Ø3 ØF4 n b2     Ø3 ØBC n c3     Ø3 Ø86 n c#3    Ø3 Ø53 n d3
    Ø3 Ø24 n d#3    Ø2 ØF6 n e3     Ø2 ØCC n f3     Ø2 ØA4 n f#3
    Ø2 Ø7E n g3     Ø2 Ø5A n g#3    Ø2 Ø38 n a3     Ø2 Ø18 n a#3
    Ø1 ØFA n b3     Ø1 ØDE n c4     Ø1 ØC3 n c#4    Ø1 ØAA n d4
    Ø1 Ø92 n d#4    Ø1 Ø7B n e4     Ø1 Ø66 n f4     Ø1 Ø52 n f#4
    Ø1 Ø3F n g4     Ø1 Ø2D n g#4    Ø1 Ø1C n a4     Ø1 ØØC n a#4 -->

SCR # 55
  ( Music - Experimental Constructs                       Ø52681-MPB )

  ( Note definitions                                                  )
    ØØ ØFD n b4     ØØ ØEF n c5     ØØ ØE1 n c#5    ØØ ØD5 n d5
    ØØ ØC9 n d#5    ØØ ØBE n e5     ØØ ØB3 n f5     ØØ ØA9 n f#5
    ØØ Ø9F n g5     ØØ Ø96 n g#5    ØØ Ø8E n a5     ØØ Ø86 n a#5
    ØØ Ø7F n b5     ØØ Ø77 n c6     ØØ Ø71 n c#6    ØØ Ø6A n d6
    ØØ Ø64 n d#6    ØØ Ø5F n e6     ØØ Ø59 n f6     ØØ Ø54 n f#6
    ØØ Ø5Ø n g6     ØØ Ø4B n g#6    ØØ Ø47 n a6     ØØ Ø43 n a#6
    ØØ Ø3F n b6     ØØ Ø3C n c7     ØØ Ø38 n c#7    ØØ Ø35 n d7
    ØØ Ø32 n d#7    ØØ Ø2F n e7     ØØ Ø2D n f7     ØØ Ø2A n f#7
    ØØ Ø28 n g7     ØØ Ø26 n g#7    ØØ Ø24 n a7     ØØ Ø22 n a#7
    ØØ Ø2Ø n b7     ØØ Ø1E n c8     ØØ Ø1C n c#8    ØØ Ø1B n d8
    ØØ Ø19 n d#8    ØØ Ø18 n e8     ØØ Ø16 n f8     ØØ Ø15 n f#8
    ØØ Ø14 n g8     ØØ Ø13 n g#8    ØØ Ø12 n a8     ØØ Ø11 n a#8
    ØØ Ø1Ø n b8     -->

SCR # 56
  ( Music - Experimental Constructs                       Ø52481-MPB )

  ( Clear all music board regs and set amplitudes                     )
  : RINIT ØFF ENABLE1 ØFF ENABLE2 3 Ø DO I CHANNEL !
      7 Ø DO Ø I SET LOOP ØB 8 DO ALV2-6 @ I SET LOOP
    ØE ØB DO Ø I SET LOOP 2 +LOOP Ø CHANNEL ! 2 VOICE !
    ALV1 @ AMP ;

  ( Voices enable/disable definitions                                 )
  : VON  EMASK1  @  ENABLE1  EMASK2  @  ENABLE2  ;
  : VOFF         ØFF  ENABLE1          ØFF  ENABLE2  ;
  -->





ok


SCR # 57
  ( Music - Experimental Constructs                       Ø52481-MPB )

  ( Rest duration definitions                                         )
  : r  <BUILDS  C,  DOES>  C@  DVAL  @  SWAP  /  Ø  DO  LOOP  ;
    64 r fr     32 r tr     16 r sr     Ø8 r er
    Ø4 r qr     Ø2 r hr     Ø1 r wr

  ( Slur duration definitions                                         )
  : sl <BUILDS C, DOES> C@ DVAL @ SWAP / Ø VON DO LOOP ;
    64 sl fs     32 sl ts     16 sl ss     Ø8 sl es
    Ø4 sl qs     Ø2 sl hs     Ø1 sl ws

  ( Dotted slur duration definitions                                  )
  : fds  DVAL  @  64  /  DUP  2  /  +  Ø  VON  DO  LOOP  ;
  : tds ts fs ;     : sds ss ts ;     : eds es ss ;
  : qds qs es ;     : hds hs qs ;     : wds ws hs ;   -->
```

```
SCR # 58
( Music Experimental Constructs                              052481-MPB )

( Note duration definitions                                            )
: d  <BUILDS C, DOES> C@   DVAL @ SWAP / 0 VON DO LOOP
                       VOFF IVAL @        0     DO LOOP ;
  64 d f    32 d t    16 d s    08 d e
  04 d q    02 d h    01 d w

( Dotted note duration definitions                                     )
: do  VOFF  IVAL  @  0  DO  LOOP  ;
: fd fds do ;     : td tds do ;     : sd sds do ;
: ed eds do ;     : qd qds do ;     : hd hds do ;
: wd wds do ;
DECIMAL  RINIT
CR  ."  Music Constructs Loaded "
;S


SCR # 59
(                                                            052481-MPB )


ok


SCR # 64
( Song - Red River Valley                                    051281-MPB )

: RR1 v4 c5 es f5 e v2 a5 v3 f5 v4 c5 v5 f3 es c4 e
   v3 g5 v5 d#4 v6 d#3 es v2 c6 v3 f#5 v5 d4 v6 d3 e v2 a5
   v3 f5 v4 b4 v6 g3 q v2 g5 v3 e5 v4 a#4 v5 c4 v6dis q
   v2 f5 v3 c5 v4 f4 v5 c4 v6 f3 qs v2 a5 v3 f5 v4 c5 es
   v2 c6 v3 a5 v4 f5 e v2 f6 qd ;
: RR2 v3dis v4dis v5dis v6dis v1 c5 v2 c5 es v1 f5 v2 f5 e
   v1 a5 v2 a5 v3 c5 v4 f3 v5 f3 es v4 c4 e v4 e4 es
   v1 g5 v2 g5 v3 a#4 v4 c4 e v1 f5 v2 f5 v3 a4 v4 f4
   v5 f3 q v1 g5 v2 g5 v3 d#5 v4 a4 v5 c4 v6 f3 es v1 f5
   v2 f5 e v1 d5 v2 d5 v3 a#4 v4 a#3 v5 a#2 v6dis es
   v1 f5 v2 f5 v3 d5 es v4 g4 v5 a#3 es v4 e4 es d4 q ;
-->


SCR # 65
( Song - Red River Valley                                    051281-MPB )

: RR3 v3dis v4dis v5dis v6dis v1 c5 v2 c5 es v1 f5 v2 f5 e
   v1 a5 v2 a5 v3 c5 v4 f3 v5 f3 es v4 c4 es v1 a5 v2 a5
   v3 f5 v6 c5 v4 d4 es v4 c4 e v1 c6 v2 c6 v3 f5 v4 c5
   v5 a4 v6dis es v5 c4 es v1 a#5 v2 a#5 v3 e5 v4 c5
   v5 g4 es v1 a5 v2 a5 v3 f5 v5 f4 e v1 g5 v2 g5 v3 e5
   v4 c5 v5 g4 v6 c4 qs v2dis v4 e4 es v3 b3 v4 f4 v5 d5
   es v1 g5 v3 a#4 v4 e4 v5 c4 q ;
: RR4 v3dis v4dis v5dis v6dis v1 c5 v2 c5 es v1 f5 v2 f5 e
   v1 a5 v2 a5 v3 c5 v4 f3 v5 f3 es v4 c4 es e4 es v1 g5
   v2 g5 v3 a#4 v4 c4 e v1 f5 v2 f5 v3 a4 v4 f4 es v4 c4 e
   v1 g5 v2 g5 v3 e5 v4 a#4 v5 c4 es v1 a5 v2 a5 v3 d#5
   v4 c5 v5 f3 e ;

-->

SCR # 66
( Song - Red River Valley                                    051281-MPB )

: RR5 v1 c6 v2 c6 v3 f5 v4 e5 v5 a#3 v6 a#2 es v1 a#5
   v2 a#5 v3 f5 v4 d5 es v6dis v5 f4 es g4 es a#4 q
   v1 d5 v2 d5 v3 g#4 v4 f4 v5 a#3 es v1 c#5 v2 c#5
   v5 b3 e v1 c5 v2 c5 v3 a4 v4 f4 v5 c4 qs v1 f5
   v2 f5 v3 c5 v4 a4 v5 c3 es e v1 a5 v2 a5 v3 f5
   v4 b4 v5 d4 v6 g3 q v1 g5 v2 g5 v3 e5 v4 a#4 v5 c4
   v6dis es e v1 f5 v2 f5 v3 a4 v4 c4 v5 f3 qs v3 a#4
   v4 d4 es es v3 a4 v4 c4 q ;
: RR6 v1 c6 v2 c6 v3 f5 v4 d#5 v5 a#3 v6 a#2 es v1 a#5
   v2 a#5 v4 d5 es v6dis v5 f4 es a4 es a#4 e v1 d5
   v2 d5 v3 g#4 v4 f4 v5 a#3 es b3 v1 c#5 v2 c#5 e ;
-->


SCR # 67
( Song - Red River Valley                                    052481-MPB )

: RR7 v1 c5 v2 c5 v3 a4 v4 f4 v5 c4 q  c3 v1 f5 v2 f5
   v3 c5 v4 a5 es e v1 a5 v2 a5 v3 f5 v4 b4 v5 d4
   v6 g3 q v5dis v6dis v1 g5 v2 g5 v3 e5 v4 a#4 es e
   v1 f5 v2 f5 v3 c5 v4 a4 v5 c4 v6 f3 es es v2 a5
   v3 f5 v4 c5 es f5 v2 c6 v3 a5 es v2 f6 v3 c6 v4 f5
   v5 a4 qd ;
: RIVER 10000 DVAL ! 256 IVAL !
                 RR1 RR2 RR3 RR4 RR5 RR2 RR3 RR4 RR5
                 RR2 RR3 RR4 RR6 RR7 RINIT ;
CR ."  Red River Valley loaded "
;S



ok
```

```
SCR # 69
( Song - Jesus Christ Superstar                              051281-MPB )

: JC1 v1 g6 v2 e6 v3 c6 v4 g5 v5 c4 v6 c3 q v3dis v4dis
   v1 e6 v2 e5 e v1 c6 v2 g5 v3 e5 v4 c5 es h v1 a6 v2 f6
   v3 c6 v4 a5 q v3dis v4dis v1 f6 v2 f5 e v1 c6 v2 a5
   v3 f5 v4 c5 es h v1 a#6 v2 f6 v3 d6 v4 a#5 q v3dis
   v4dis v1 g6 v2 g5 e v1 a#6 v2 a#5 e v1 a6 v2 f6 v3 c6
   v4 a5 q v3dis v4dis v1 g6 v2 g5 e v1 f6 v2 f5 e
   v1 g6 v2 e6 v3 c6 v4 g5 q v3dis v4dis v1 e6 v2 e5 e
   v1 c6 v2 g5 v3 e5 v4 c5 es q v1dis v2dis v3dis v4dis
   v5dis v6 g3 s e s ;
: JC2 v6dis v2 g5 v3 e5 v4 c4 v1 a#5 e c6 e ss v4dis s
   v1 a#5 v4 g3 e c4 v1 c6 e a#5 e c6 e a#5 e ;
: JCA c6 v3 d#5 v4 d#4 e v2dis v3dis v1 a#5 e g5 ss v4dis
   s v1 d#5 v2 a#4 v3 g4 v4 a#3 es d#3 gs g3 es a#3 e ;
-->


SCR # 70
( Song - Jesus Christ Superstar                              051281-MPB )

: JC3 v1 c6 v2 a5 v3 d#5 v4 f3 e v1 a#5 ss v4dis s v1 c6
   e v1 b5 v4 f3 e v1 c6 e b5 e c6 e b5 e v1 c6 v2dis
   v3 e5 v4 c4 e v1 a#5 ss v3dis v4dis s v1 g5 e
   v1 e5 v2 a#4 v3 g4 v4 c4 es es es g3 es es e ;
: JC4 v1 d#5 v2 a5 v3 f5 v4 f4 e e ss v4dis s v4 s s
   v4 c4 e e f3 e v1 c6 e v1 d#6 v2 a#5 v3 e5 v4 c4 e
   v1 e6 e c6 ss v4dis s a#5 v4 es v1 c6 v2 v3 es v4 g3
   c4 es g3 e ;
: JC5 v1 a#5 v2 g5 v3 e5 v4 c4 e v1 c6 e ss v4dis s v4
v1 a#5 e e c6 e a#5 e c6 e d#5 v2 a#5 v3 g5 v4
   d#3 e f6 e g6 ss v2dis v3dis v4dis s v1 f6 v2 v3 v4
   a#3 es d#3 e v1 c6 v2 g5 v3 d#5 qd ; -->


SCR # 71
( Song - Jesus Christ Superstar                              051281-MPB )

: JC6 v2 a5 v3 d#5 v4 f3 v1 c6 e e ss v4dis s v4 c4 e f4
   e e v1 a#5 e c6 e d#6 v2 a#5 v3 e5 v4 c4 e v1 c6 e
   a#5 v2 f5 v3 d5 v4 g3 e v1 c6 e d#6 v2 a#5 v3 e5 v4 c4 e
   v1 c6 v2 g5 v3 e5 es v4 g3 q v1dis v2dis v3dis ;
: JC7 c4 es v1 g5 s s a#5 v2 f5 v3 d5 s e5 v1 c6 v2 g5 e
   ss h v1dis v2dis v3dis v4 f4 es v1 g5 s s a#5 v2 g5
   v3 d#5 s v1 c6 v2 a5 e ss qs v4 c4 q v1dis v2dis v3dis
   f4 es v1 a5 s s v2 f5 s v2dis v1 c6 e f6 v2 c6 v3 a5
   ss v4 c4 qs f3 q ;
-->


SCR # 72
( Song - Jesus Christ Superstar                              051581-MPB )

: JC8 v1dis v2dis v3dis v4 c4 es v1 g6 s s f6 v2 c6 v3 a5 s
   v2dis v3dis v1 d#6 e c6 v2 g5 v3 e5 ss qs v4 g3 es e
   v1dis v2dis v3dis c4 es v1 g5 s s a#5 v2 f5 v3 d5 s e5
   v1 c6 v2 g5 e ss qs v4 g3 es c4 e v1dis v2dis v3dis
   f4 es v1 a5 s s v2 f5 s v2dis v1 c6 e f6 v2 c6 v3 a5 ss
   v4 c4 qs f3 es c4 e ;
: JC9 v1dis v2dis v3dis v4 f4 es v1 f6 s f#6 s f6 v2 c6
   v3 a5 s v2dis v3dis v1 d#6 e f6 v2 c6 v3 a5 ss v4 d#4
   qs c4 es f3 e v1dis v2dis v3dis c4 es v1 f#6 s s f6
   v2 c6 v3 a5 s v2dis v3dis v1 d#6 e c6 v2 g5 v3 e5 ss
   v4 a#3 qs g5 q ;
-->


SCR # 73
( Song - Jesus Christ Superstar                              051281-MPB )

: JC10 v1 g5 v2 e5 v3 c5 v4 g4 v5 c4 v6 c3 q v2dis v3dis v4dis
   v1 e5 e c5 v2 g4 v3 e4 es h v1 a5 v2 f5 v3 c5 v4 a4 v5 f4
   v6 f3 q v2dis v3dis v4dis v1 f5 e c5 v2 a4 es h f5 v1 a#5
   v3 d5 a#4 v5 a#3 v6 a#2 q v2dis v3dis v4dis v1 g5 e a#5 e
   a5 v2 f5 v3 c5 v4 a4 v5 f4 v6 f3 q v2dis v3dis v4dis v1 g5
   e f5 e g5 v2 e5 v3 c5 v4 g4 v5 v5 c4 v6 c3 q v2dis v3dis
   v4dis v1 e5 e c5 v2 g4 v3 e4 es es v6dis v5 g3 qs e v5dis ;
-->


SCR # 74
( Song - Jesus Christ Superstar                              052481-MPB )

: JC11 v1 g5 v2 e5 v3 c5 v4 c4 q v2dis v3dis v1 e5 e c5 v2 g4
   v3 e4 v4 c4 es qs g3 es c4 e f3 v1 a5 v2 f5 v3 c5 q v2dis
   v3dis v1 f5 e c5 v2 a4 v3 f4 v4 f3 es qs es a3 e a#3 v1 a#5
   v2 f5 v3 d5 q v2dis v3dis v1 g5 e a#5 e a5 v2 f5 v3 c5 v4 f3
   q v2dis v3dis v1 g5 v4 g3 e a3 v1 f5 e g5 v2 e5 v3 c5 v4 c4
   q v2dis v3dis v1 e5 e c5 v2 g4 v3 e4 es es v4 g3 qs e ;
: JCFA ALV1 @ 1 - ALV1 ! ALV2-6 @ 1 - ALV2-6 ! v1 ALV1 @ AMP
   v2 ALV2-6 @ AMP v3 ALV2-6 @ AMP v4 ALV2-6 @ AMP ;
: JCEND 08 ALV1 ! 6 ALV2-6 ! ;
: SUPERSTAR 11000 DVAL ! 128 IVAL !
   JC1 JC2 JCA JC3 JC2 JCA JC4 JC5 JC6 JC7
   JC8 JC9 JC10 JC10 JC2 JCA JC3 JC2 JCA JC4 JC5 JC6 JC7 JC8 JC9
   JC11 JCFA JC11 JCFA JC11 JCFA JCFA JC11 JCEND RINIT ;
CR ."  Jesus Christ Superstar loaded " ;S

ok
```

OPTIMIZING DICTIONARY SEARCHES

Paul van der Eijk
5480 Wisconsin Avenue, #1128
Chevy Chase, MD 20015
(301) 656-2772

Recently, I finished the implementation of fig-FORTH on my Radio Shack model II. I must admit that I did not follow the FIG model precisely; some high level definitions were recoded in assembler to increase their speed. For example, sign extraction in the divide and multiply words gives an execution time improvement of a factor two. These improvements are predictable and probably implemented many times already.

One deviation from the FIG model I want to share with you is the structure of the dictionary.

In the FIG model, the Link Field Address is stored after the last character of the name. When (FIND) searches the dictionary for an entry, the lengths of the strings are compared. If the comparison fails, and this happens a lot, the characters stored are scanned for a high bit in the last character. When the scan stops at the last character, we know the address of the LFA, because it follows the last character. It will be obvious that the time spent on searching for the LFA will be linear with the average numbers of characters stored for an entry. One way to get around scanning is adding an additional byte in every dictionary entry, indicating the actual number of characters stored. Another approach was taken by Robert Smith, see FORTH DIMENSIONS, Vol. 1, No. 5.

The structure I implemented puts the LFA in front of the Name Field Address. When (FIND) stores the address of the NFA in a machine register, a search for the LFA is not necessary because it precedes the NFA directly. In addition, the characters of the entry can be stored in normal order, which makes changing ID. unnecessary.

The new dictionary structure can improve compilation speed substantially. An application program 70 screens long took 210 seconds to compile; the new dictionary structure reduced the compilation time to 98 seconds.

To implement the new dictionary structure, the following words have to be rewritten:

CREATE VOCABULARY LFA NFA PFA .
(FIND) has to be rewritten as well, but is not given here because it is machine dependent.

```
( LFA preceeds NFA  1 of 2:
HEX
: CREATE  -FIND  IF  DROP  NFA  ID.  4  MESSAGE SPACE THEN
                               ( check unique in CURRENT and CONTEXT )
  HERE  C@  WIDTH  @  MIN  >R      ( save number of chars stored )
  HERE  0A0  TOGGLE  HERE  R  +  DUP  080  TOGGLE
                                          ( smudge and delimiter bits )
  DUP  2+  R  1+  -CMOVE          ( move entry down to insert LFA )
  LATEST  HERE  !   HERE  2+  CURRENT  @  !
  R>  3  +  ALLOT   HERE  2+   ,  ;

: NFA  3  -  -1  TRAVERSE  ;
: LFA  NFA  2  -  ;
: PFA  1  TRAVERSE  3  +  ;

--
```

```
( LFA preceeds NFA  2 of 2:        Paul van der Eijk april-12-1981 )
: VOCABULARY  <BUILDS  CURRENT  @  2+  ,  0A081   ,
  HERE  VOC-LINK  @  ,  VOC-LINK  !
  DOES>  CONTEXT  !  ;


( the following change in -FIND speeds up dictionary searches    )
( in case the CURRENT and CONTEXT vocabularies are the same.     )
( the change is not necessary for the new dictionary structure )

: -FIND  BL  WORD  HERE  CONTEXT  @  @  (FIND)  DUP  0=
  IF    DROP   LATEST  CONTEXT  @  @  OVER  -
        IF   HERE  SWAP  (FIND)
        ELSE DROP  0
  THEN  THEN  ;
DECIMAL     ;S
```

MEETING

POTOMAC FORTH INTEREST GROUP MEETING

Program was presented by Paul van der Eijk on IQS - An Interactive Query System. He described this system which lets the user create, edit, search and list a file without writing a program.

The next meeting is Tuesday, Aug. 4, 1981 at Lee Center, Lee Hiahway at Lexington St., Arlington, Virginia from 7:00 - 9:00pm.

TRACING COLON-DEFINITIONS

Paul van der Eijk
5480 Wisconsin Avenue, #1128
Chevy Chase, MD 20015
(301) 656-2772

This short article describes a few simple words to trace colon definitions. When I am completely lost trying to find a bug in a FORTH program, I use colon tracing to get a print-out of all words executed together with a few parameters on the data-stack. Such a print-out is often enough to spot the bug; in addition, it gives some insight how many times certain words are executed which can help to improve the execution time of a program.

How it works:

A technique to trace colon definitions is to insert a tracing word directly after the colon.

i.e., : TEST T1 T2 ; TEST can be traced by having a definition compiled as if it were:

: TEST (TRACE) T1 T2;

When (TRACE) executes, the address of the word following it is on the return stack. Subtracting two from this address will give the parameter field address, from which we can reach the name field address using the word NFA. In order to enable/disable the trace ouput, the variable TFLAG is used; a non-zero value will enable the output and a zero value will suppress the trace output.

The insertion of the (TRACE) word can be automated if we redefine the definition of the colon.

The colon is redefined to insert the runtime procedure for the colon followed by the address of (TRACE).

Note that the address of the colon runtime procedure is obtained by taking it from the code field address of the word (TRACE).

Improvements:

1. If we save in (TRACE) the value of the variable OUT and direct output to the line-printer, words doing formatted terminal output can be debugged effectively.

2. A variable TRACE is introduced to control the insertion of the word (TRACE) in the new definition for the colon.

If the value of TRACE equals zero, (TRACE) is not inserted, if the value is non-zero (TRACE) will be inserted.

This enables tracing code to be inserted in a selective way by changing the value of TRACE preceding a colon definition.

i.e.:

0 TRACE ! : TEST1 T11 T12 ; (TEST1 will not be traced)

1 TRACE ! : TEST2 T21 T22 ; (TEST 2 can be traced)

```
( trace colon words:           Paul van der Eijk  april-12-1981 )
FORTH  DEFINITIONS
0 VARIABLE TFLAG                            (controls trace output )
: (TRACE)    ( give trace output. to be inserted as first word )
  TFLAG @                                 ( trace output if non-zero )
  IF  CR  R  2  -  NFA  DUP  ID.     ( back to PFA NFA for name )
      C@  31 AND  32  SWAP  -  SPACES       ( add spaces to name )
      -2  4  DO  SP@  I  +  @  8  .R  -2  +LOOP    ( show stack )
  THEN ;
: :                    ( redefined to insert trace word after colon )
  ?EXEC  !CSP  CURRENT  @  CONTEXT  !  CREATE
  '  (TRACE)  CFA  DUP  @  HERE  2  -  !  ,  ]  ;  IMMEDIATE

( example: trace following use of !  and C! )
:  !  !  ;
:  C!  C!  ;                                          ;S
```

MEETING

NEW YORK CHAPTER

First meeting of the New York Chapter was held on June 23, 1981. There were five FIG members and one non-FIG person in attendance. The second meeting is scheduled for August 25, 1981 and subsequent meetings every other month.

FORTH, Inc. NEWS

We're Growing

FORTH, Inc. expects to double its staff within the next year to accommodate increased product demand and applications programming. (See job openings listed elsewhere in this publication.)

The latest addition to our staff is programmer Charles Curley. Curley is a former freelance writer and programmer who edits and publishes the Ohio Scientific Users' Newsletter.

"I put FIG FORTH up in my own Ohio Scientific C2-8P DF and liked it," he comments, "but I wanted to learn FORTH systematically, and I figured this was the best place to do that. At FORTH, I get paid to do what I like to do."

Other News

President Elizabeth D. Rather was a member of a panel on programming languages for small computers at the NCC Convention. She was featured in both Computerworld and Computer Business News.

Programmer Mike LaManna has relocated to Long Island, New York, and is working on the 68000 polyFORTH. It should be available midsummer.

PolyFORTH Palo Alto Users Groups Starting

Dr. C. H. Ting has volunteered to Chair the Palo Alto Thread of the FORTH Users Group for the first three months. Anyone interested in joining the Users Group may contact Dr. Ting at Lockheed Missiles and Space Corp., (408) 742-1101 or Al Krever at FORTH, Inc. (213) 372-8493.

Recent Applications

FORTH, Inc. has produced a computer numerical control program for L & F Industries' rotating longitudinal-stretch forming machine. This 80-foot-long, three-story-high giant weighs over a million pounds and pulls 750 tons. It is used to form, stretch, bend and stretch wrap aluminum, steel and titanium sheet metal or extrusion parts (typically panels used in C5A-sized aircraft).

An LSI-11 detects the yield point of the metal and maintains a pre-set stress as the operator directs the initial operation; it then takes over full control and manufactures identical production parts. This computer program, written in polyFORTH, coordinates the motion of nine simultaneously moving servo-controlled axes with a resolution of .008". The system also displays on a CRT the position of all axes and a graph of the stress curve showing the yield point of the metal. Mike La Manna, Jim Dewey and Gary Friedlander were involved in the project.

Starting FORTH Preprints Available

A few unsigned preprints of Starting FORTH are available now for $30 (plus 6% state tax). The Prentice-Hall edition will be available in book stores September 8. To order a preprint, send a check to Winnie Shows at FORTH, Inc., 2309 Pacific Coast Hwy., Hermosa Beach, CA 90254 or you may call her at (213) 372-8493 with a VISA or MASTERCHARGE number.

FORTH, Inc. Seminars, Workshops, Classes

Location	**Seminar**	**Workshop**
Chicago	August 4	August 5
Boston	August 6	August 7
Boulder, CO	September 1	September 4
Los Angeles	October 15	October 16
San Diego	October 22	October 23

An introductory class in polyFORTH programming will be offered August 10-14 at FORTH, Inc. Call Kris Cramer for details (213) 372-8493.

FORTH VENDORS

The following vendors have versions of FORTH available or are consultants:

ALPHA MICRO
Professional Management Services
724 Arastradero Rd. #109
Palo Alto, CA 94306
(415) 858-2218

Sierra Computer Co.
617 Mark NE
Albuquerque, NM 87123

APPLE
IUS (Cap'n Software)
281 Arlington Avenue
Berkeley, CA 94704
(415) 525-9452

George Lyons
280 Henderson St.
Jersey City, NJ 07302
(201) 451-2905

MicroMotion
12077 Wilshire Blvd. #506
Los Angeles, CA 90025
(213) 821-4340

CROSS COMPILERS
Nautilus Systems
P.O. Box 1098
Santa Cruz, CA 95061
(408) 475-7461

polyFORTH
FORTH, Inc.
2309 Pacific Coast Hwy.
Hermosa Beach, CA 90254
(213) 372-8493

LYNX
3301 Ocean Park #301
Santa Monica, CA 90405
(213) 450-2466

M & B Design
820 Sweetbay Drive
Sunnyvale, CA 94086

Micropolis
Shaw Labs, Ltd.
P. O. Box 3471
Hayward, CA 94540
(415) 276-6050

North Star
The Software Works, Inc.
P. O. Box 4386
Mountain View, CA 94040
(408) 736-4938

OSI
Consumer Computers
8907 LaMesa Blvd.
LaMesa, CA 92041
(714) 698-8088

Software Federation
44 University Dr.
Arlington Heights, IL 60004
(312) 259-1355

Technical Products Co.
P. O. Box 12983
Gainsville, FL 32604
(904) 372-8439

Tom Zimmer
292 Falcato Dr.
Milpitas, CA 95035

6800 & 6809
Kenyon Microsystems
3350 Walnut Blvd.
Houston, TX 77042
(713) 978-6933

PDP-11
Laboratory Software Systems, Inc.
3634 Mandeville Canyon Rd.
Los Angeles, CA 90049
(213) 472-6995

John S. James
P. O. Box 348
Berkeley, CA 94701

TRS-80
Miller Microcomputer Services
61 Lake Shore Rd.
Natick, MA 01760
(617) 653-6136

The Software Farm
P. O. Box 2304
Reston, VA 22090

Sirius Systems
7528 Oak Ridge Hwy.
Knoxville, TN 37921
(615) 693-6583

KIM
Eric C. Rehnke
540 S. Ranch View Circle #61
Anaheim Hills, CA 92087

8080/Z80/CP/M
Laboratory Microsystems
4147 Beethoven St.
Los Angeles, CA 90066
(213) 390-9292

Mitchell E. Timin Engineering Co.
9575 Genesse Ave. #E-2
San Diego, CA 92121
(714) 455-9008

Consultant
Henry Laxen
1259 Cornell
Berkeley, CA 94706
(415) 525-8582

Application Packages
InnoSys
2150 Shattuck Avenue
Berkeley, CA 94704
(415) 843-8114

Decision Resources Corp.
28203 Ridgefern Ct.
Rancho Palo Verde, CA 90274
(213) 377-3533

Firmware, Boards and Machines
Datricon
7911 NE 33rd Dr.
Portland, OR 97211
(503) 284-8277

Forward Technology
2595 Martin Avenue
Santa Clara, CA 95050
(408) 293-8993

Rockwell International
Microelectronics Devices
P.O. Box 3669
Anaheim, CA 92803
(714) 632-2862

Zendex Corp.
6398 Dougherty Rd.
Dublin, CA 94566

Variety of FORTH Products
Interactive Computer Systems, Inc.
6403 Di Marco Rd.
Tampa, FL 33614

Mountain View Press
P. O. Box 4656
Mountain View, CA 94040
(415) 961-4103

Supersoft Associates
P.O. Box 1628
Champaign, IL 61820
(217) 359-2112

Consultants
Creative Solutions, Inc.
4801 Randolph Rd.
Rockville, MD 20852

Dave Boulton
581 Oakridge Dr.
Redwood City, CA 94062
(415) 368-3257

Elmer W. Fittery
110 Mc Gregor Avenue
Mt. Arlington, NJ 07856
(213) 663-1580

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

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

DECOMPILER FOR SYN-FORTH

SYN-1 User's Group
PO Box 315
Chico, CA 95927

The following decompiler works very well except that because INTERPRET is not remembered by ;S nor (;CODE-) nor QUIT, this FORTH decompiles loop !

```
1 LIST 2 LIST 3 LIST
SCR#1
  0 ( DECOMPILER... ) HEX
  1
  2 VOCABULARY UTIL IMMEDIATE  FORTH DEFINITIONS
  3
  4 : PICK 2 * SP@ + @ ;        UTIL  DEFINITIONS
  5
  6 : TEST= ( PFA F PFA? -> PFA F )  3 PICK = OR ;
  7
  8 : TEST.END ( @CFA -> @CFA F )  DUP @ 2+ 0
  9              ' ;S      TEST=
 10              ' (;CODE) TEST=
 11              ' QUIT    TEST=
 12              ?TERMINAL OR SWAP DROP 0= ;
 13
 14
 15 -->

SCR # 2
  0 ( ...DECOMPILER... )
  1
  2 : INCREMENT ( @CFA -> @CFA+ ) DUP 2+ SWAP @ 2+
  3          0 ' COMPILE TEST= IF DROP DUP @ 2+ NFA ID. 2+ SPACE
  4     ELSE 0 ' CLIT    TEST= IF DROP DUP C@ HEX . 1+ SPACE
  5     ELSE 0 ' (.")    TEST= IF DROP DUP COUNT DUP ROT SWAP
  6                                TYPE + 1+ 22 EMIT SPACE
  7     ELSE 0 ' LIT     TEST= DUP IF HEX THEN
  8            ' BRANCH  TEST=
  9            ' OBRANCH TEST=
 10            ' (LOOP)  TEST= IF DROP DUP @ . SPACE 2+ ELSE DROP
 11     THEN THEN THEN DECIMAL THEN ;
 12
 13
 14
 15 -->

SCR # 3
  0 ( ...DECOMPILER... ) FORTH DEFINITIONS HEX
  1
  2 : DECOMPILE ( PFA -> NFA OF NEXT WORD ) UTIL
  3       HEX DUP 4 .R DUP CFA @ 5 .R DECIMAL
  4       DUP CFA @ 803 =
  5       IF ."  : " DUP NFA ID. SPACE DUP
  6          BEGIN DUP @ 2+ NFA DUP C@ 40 AND IF ." [COMPILE] " THEN
  7                ID. TEST.END
  8          WHILE INCREMENT
  9          REPEAT
 10          DROP DUP NFA @ 40 AND IF ." IMMEDIATE " THEN
 11       ELSE ."    " DUP NFA ID.
 12       THEN LFA @ CR CR ;
 13
 14 : DECOMPILE.ALL ( PFA -> PFA )  CR
 15      BEGIN DECOMPILE DUP PFA SWAP 0= ?TERMINAL OR UNTIL CR ;
OK

EXAMPLES
' . DECOMPILE.ALL
1A17  803 : .  S->D D. ;S
1A07  803 : .R  >R S->D R> D.R ;S
19F8  803 : D.  0 D.R SPACE ;S
19D5  803 : D.R  >R SWAP OVER DABS <# #S SIGN #> R> OVER - SPACES TYPE ;S
19BD  803 : #S  # OVER OVER OR 0= OBRANCH -12  ;S
1995  803 : #  BASE @ M/MOL ROT CLIT 9  OVER < OBRANCH 7  CLIT 7  + CLIT 30  + H
OLD ;S
1980  803 : SIGN  ROT 0< OBRANCH 7  CLIT 2D  HOLD ;S
1967  803 : #>  DROP DROP HLD @ PAD OVER - ;S
1958  803 : <#  PAD HLD ! ;S
193B  803 : SPACES  0 MAX -DUP OBRANCH 12  0 (DO) SPACE (LOOP) -4  ;S
192A  803 : WHILE  [COMPILE] IF 2+ ;S IMMEDIATE
1908  803 : ELSE  2 ?PAIRS COMPILE BRANCH  HERE 0 , SWAP 2 [COMPILE] ENDIF 2 ;S
IMMEDIATE
18F1  803 : IF  COMPILE OBRANCH  HERE 0 , 2 ;S IMMEDIATE
18D8  803 : REPEAT  >R >R [COMPILE] AGAIN R> R> 2 - [COMPILE] ENDIF IMMEDIATE

OK
' WHILE DECOMPILE 192A  803 : WHILE [COMPILE] IF 2+ ;S IMMEDIATE

OK


4 OK
LIST
```

```
SCR # 4
  0  COLON DEFINITIONS DECOMPILER FOR SYM-1
  1  MICHEL DESSAINTES    MARCH 22, 1981
  2
  3  USE : ' WORD.TO.DECOMPILE DECOMPILE
  4   OR : ' FIRST.WORD.TO.DECOM DECOMPILE.ALL
  5
  6  OUTPUT : PFA CFA.CONTENT : WORD.TO.DECOMP ...
  7
  8  DISPLACEMENT ARE IN DECIMAL (BRANCH, OBRANCH, LIT,...)
  9  OTHER LITTERAL ARE IN HEXADECIMAL (LIT, CLIT,...)
 10
 11  POSSIBLE EXTENSIONS : REPLACE (LOOP), (BRANCH),...
 12                         BY LOOP, BRANCH,...
 13                        JUSTIFY IF ... ELSE ... ELSE, ...
 14                        DECOMPILE VARIABLE, VOCABULARY, ...
 15
OK
```

ENGLISH FORTH APPLICATION

Golden River company has been using FORTH for the RCA 1802 for the last three years, to fill a need for a low cost development and prototyping tool with potential for being used at remote sites where power is not easily available.

The most interesting concept in the equipment is it uses 32K of dynamic RAM as storage space for up to 30 screens of source FORTH code. The equipment is designed with low power in mind and is normally used like an electric car--it is usually kept connected to an AC source, although it has nine-days battery life and can be used remotely.

The product is currently being shipped in Europe and will be introduced in the U.S. market through Golden River Corporation, 7315 Reddfield Court, Falls Church, VA 22043.

For more information, contact:

Golden River Company, Ltd.
Churchill Road
Bicester, Oxfordshire OX6 7XT
England
Phone: Bicester (08692) 44551
Telex: 83147 VIAOR G 'GRIVER'

GET READY!

FORML's COMING!

NEW PRODUCT ANNOUNCEMENT FORMAT

In the interests of comparison uniformity and completeness of data in new product announcements, FORTH DIMENSIONS requests that all future new product announcements use the following format:

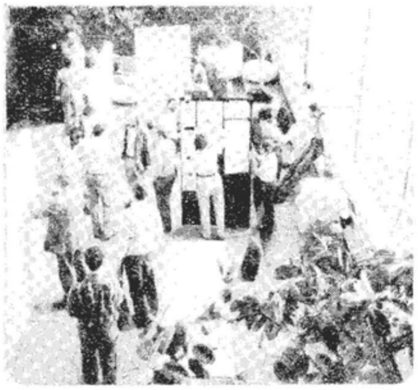
1. Vendor Name (company)
2. Vendor mailing address
3. Vendor street address if PO Box. Used as mailing address. For reference file.
4. Vendor area code and telephone number
5. Person to contact
6. Product name
7. Brief description of product uses/features
8. List of extras included (editor, assembler, data base, games, etc.)
9. List of machines product runs on
10. Memory requirements
11. Number of pages in manual
12. Tell what manual covers
13. Indicate whether or not manual is available for separate purchase
14. If manual is available, indicate separate purchase price and whether or not manual price is credited towards later purchase
15. Form product is shipped in (must be diskette or ROM--no RAM only or tape systems)
16. Approximate number of product shipments to date (product must have active installations as of writing--no unreleased products)
17. Product price
18. What price includes (shipping, tax, etc.)
19. Vendor warranties, post sale support, etc.
20. Order turn-around time

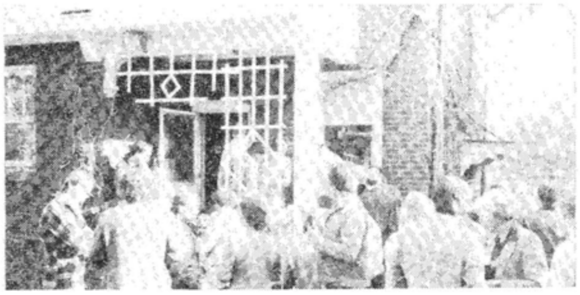
MEETINGS/EVENTS ANNOUNCEMENT FORMAT

In order to have uniformity and insure complete information in all meeting and special event announcements, FORTH DIMENSIONS requests that you use the following format:

1. WHO is holding the event (organization, club, etc.)
2. WHAT is being held (describe activity, speakers' names, etc.)
3. WHEN is it being held (days, times, etc.; please indicate if it is a repetitive event--monthly meeting etc.)
4. WHERE is it being held (be as complete as possible--room number, etc.)
5. WHY is it being held (purpose, objectives, etc.)
6. REMARKS and SPECIAL NOTES (is there a fee, are meals/refreshements being provided, dress, tools, special requirements, pre-requisites, etc.)
7. PERSON TO CONTACT
8. PHONE NUMBER/ADDRESS (include area codes, times to call and give work and home numbers in case we need clarification)

FORTH STANDARDS CONFERENCE
ROCHESTER, NY - SPRING 1981





1981 FORML CONFERENCE

Asilomar, California   November 25-27, 1981

ATTENDEE REGISTRATION FORM

Conference Purpose:	The 1981 FORML (FORTH Modification Laboratory) is an advanced seminar for the presentation of FORTH language papers and discussions. It _is not_ intended for beginning or casual FORTH programmers.
Attendee Selection Priority:	The FORML Conference is limited to 60 FORTH programmers (approx. 30 family and other non-participants accommodations are also available). The priority for selection of attendees is:

1st - Paper presentors who send in their 100-word abstract by the deadline of September 1, 1981.*

2nd - Poster presentors who send in their 100-word abstract by the deadline of September 1, 1981.*

3rd - FORTH programmers who wish to attend only. Depending upon the response of paper and poster sessions there may or may not be room for non-presentors.*

*The FORML Conference Referees will make the final decisions on paper/poster presentors which will in effect determine attendance and priority positions.

Registration Form, Complete and return to:
FORML
PO Box 51351
Palo Alto, CA
94303

NAME______________________________

ADDRESS______________________________

CITY____________ STATE_____ ZIP_____ COUNTRY_______

PHONE (Day)_______________(Evening)_______________

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

Types of CPU's and/or computers:____________________

I have authored the following papers and/or articles about FORTH:

____

I expect to: ____ present a paper, ____ present a poster session

____ chair a section, ____ non-presentor

My topic will be:______________________________

Accommodations Desired:

Rooms at Asilomar include meals (including a huge Thanksgiving) and the price of the Proceedings is included in participant costs.

____ Myself ____ Double $110.00 ____ Single $150.00

____ Wife/Husband/Friend ($85.00 for room and meals)

I will arrive the afternoon or night before, please reserve a room for: #____ on Tuesday, November 24 @ ____ $35.00 double or ____ $47.00 single

FOR MORE INFORMATION CALL: ROY MARTENS (415) 962-8653

LATE NEWS

BURKLUND & ASSOCIATES
3903 Carolyn Ave
Fairfax, VA 22031
(703)273-5663

Mr. Roy C. Martens
Forth Interest Group
P.O. Box 1105
San Carlos, CA 94070

June 29, 1981

Sirs:

Tsk, Tsk, Tsk! You really did it this time! Tsk, Tsk, Tsk. The product review of Timin Eng. version of FORTH is totally beside the point...what you did, was crucify one of the finest versions of FORTH currrently available on the market, namely the Laboratory Microsystems, Z-80 FORTH.

I bought Laboratory Microsystems Z-80 Forth about 3 months ago, and was ecstatic with what I had bought for a paltry $50. When I read your product review, I tried the same definitions on my 4 MHZ system and found that all times were approximately 2-5% less than your comparative data for Timins 8080 version...therefore with the differing CPU clock rates of 4MHz for my and 6Mhz for Timins systems on which the Laboratory Microsystems Z-80 versions were compared (how convenient is was tried on Mr. Timins systems) the Z-80 version should reflect benchmark times approaching 30% better than those cited in the comparison test. I would have thought that FORTH DIMENSIONS would have staff expertise of a bit higher quality than that reflected in the product review article.

As for the tip-toeing disclaimers via the Editors Comment... hey, it just won't wash!

I think that FORTH DIMENSIONS owes a very large apology to LABORATORY MICROSYSTEMS, and at least a full page of space to try to counter the damage you have done to Z-80 FORTH; or will you allow the old adage that "the truth never catches up to the lie", prevail? FORTH DIMENSIONS...Shame! Shame! Shame!

Sincerely,

[illegible]

Glenn A. Burklund

Publisher's Comment: The following letter was received in feference to a Product Review by C. H. Ting in FORTH DINENSIONS, III/1, page 11-12, which compared some bench marks between CP/M FORTH from Timin Engineering and Z-80 FORTH from Laboratory Microsystems. We are printing this letter in its entirety for several reasons: to correct any unintentional damage to Laboratory Microsystems; to ask our members whether they desire comparisions between FORTH and other languages and between FORTH products; if we are to do comparisions then it will have to be by volunteers since we do not have a staff, it then becomes a problem of who and how. Any volunteers?

How to form a FIG Chapter:

1. You decide on a time and place for the first meeting in your area. (Allow at least 8 weeks for steps 2 and 3.)

2. Send FIG a meeting announcement on one side of 8-1/2 x 11 paper (one copy is enough). Also send list of ZIP numbers that you want mailed to (use first three digits if it works for you).

3. FIG will print, address and mail to members with the ZIP's you want from San Carlos, CA.

4. When you've had your first meeting with 5 or more attendees then FIG will provide you with names in your area. You have to tell us when you have 5 or more.

Northern California

4th Sat	FIG Monthly Meeting, 1:00 p.m., at Southland Shopping Ctr., Hayward, CA. FORML Workshop at 10:00 a.m.

Southern California

Los Angeles

4th Sat	FIG Meeting, 11:00 a.m., Allstate Savings, 8800 So. Sepulveda, L.A. Call Phillip Wasson, (213) 649-1428.

Orange County

3rd Sat	FIG Meeting, 12:00 noon, Fullerton Savings, 18020 Brockhorst, Fountain Valley, CA. (714) 896-2016.

San Diego

Thur	FIG Meeting, 12:00 noon. Call Guy Kelly, (714) 268-3100, x 4784 for site.

Northwest

Seattle	Chuck Pliske or Dwight Vandenburg, (206) 542-8370.

Oregon

2nd Sat	FIG Meeting, 1:00 pm, Computers & Things, 3460 SW 185th "D", Aloha, Eric Smith, (503) 642-1234.

New England

Boston

1st Wed	FIG Meeting, 7:00 p.m., Mitre Corp., Cafeteria, Bedford, MA. Call Bob Demrow, (617) 389-6400, x198.

Boston

3rd Wed	MMSFORTH Users Group, 7:00 p.m., Cochituate, MA. Call Dick Miller, (617) 653-6136 for site.

Southwest

Phoenix	Peter Bates at (602) 996-8398.

Tulsa

3rd Tues	FIG Meeting, 7:30 p.m., The Computer Store, 4343 So. Peoria, Tulsa, OK. Call Bob Giles, (918) 599-9304 or Art Gorski, (918) 743-0113.
Texas	Jeff Lewis, (713) 719-3320 or John Earls, (214) 661-2928 or Dwayne Gustaus, (817) 387-6976. John Hastings (512) 835-1918.

Mid Atlantic

Potomac	Paul van der Eijk, (703) 354-7443 or Joel Shprentz, (703) 437-9218.
New York	Tom Jung, (212) 746-4062.

Midwest

Detroit	Dean Vieau, (313) 493-5105.

Foreign

England	FORTH Interest Group, c/o 38, Worsley Road, Frimley, Camberley, Surrey, GU16 5AU, England
Japan	FORTH Interest Group, Baba-bldg. 8F, 3-23-8, Nishi-Shimbashi, Minato-ku, Toyko, 105 Japan.

Canada

Quebec	Gilles Paillard, (418) 871-1960 or 643-2561.

West Germany

Wolf Gervert, Roter Hahn 29, D-2 Hamburg 72, West Germany,(040) 644-3985.

FORTH INTEREST GROUP
MAIL ORDER

		USA	FOREIGN AIR
☐	Membership in FORTH INTEREST GROUP and Volume III (6 issues) of FORTH DIMENSIONS.	$15	$27
☐	Volume II of FORTH DIMENSIONS (6 issues)	$15	$18
☐	Volume I of FORTH DIMENSIONS (6 issues)	$15	$18
☐	fig-FORTH Installation Manual, containing the language model of fig-FORTH, a complete glossary, memory map and installation instructions	$15	$18
☐	Assembly Language Source Listings of fig-FORTH for specific CPU's and machines. The above manual is required for installation. Check appropriate box(es). **Price per each.** ☐ 1802 ☐ 6502 ☐ 6800 ☐ 6809 ☐ 8080 ☐ 8086/8088 ☐ 9900 ☐ APPLE II ☐ PACE ☐ ALPHA MICRO ☐ PDP-11 ☐ NOVA	$15	$18
☐	PROCEEDINGS 1980 FORML (FORTH Modification Lab) Conference	$25	$35
☐	PROCEEDINGS Spring 1981 FORTH Standards Team Conference	$25	$35
☐	FORTH-79 Standard, a publication of the FORTH Standards Team	$15	$18
☐	Using FORTH, by FORTH, Inc. This is the best users manual.	$25	$32
☐	Kitt Peak Primer, by Stevens. An indepth self-study primer.	$25	$35
☐	BYTE Magazine Reprints of FORTH articles, 8/80 to 4/81	$ 5	$10
☐	FIG T-shirts: ☐ Small ☐ Medium ☐ Large ☐ X-Large	$10	$12
☐	· Poster/1981 Calendar, Aug 1980 BYTE cover, 18 x 22″	$ 3	$ 5
☐	FORTH Programmer's Reference Card. If ordered separately, send a stamped, addressed envelope.	FREE	
	TOTAL	$________	

NAME________________________MAIL STOP/APT____________

ORGANIZATION____________________(If company address)

ADDRESS____________________________________

CITY________________STATE____ZIP_____COUNTRY__________

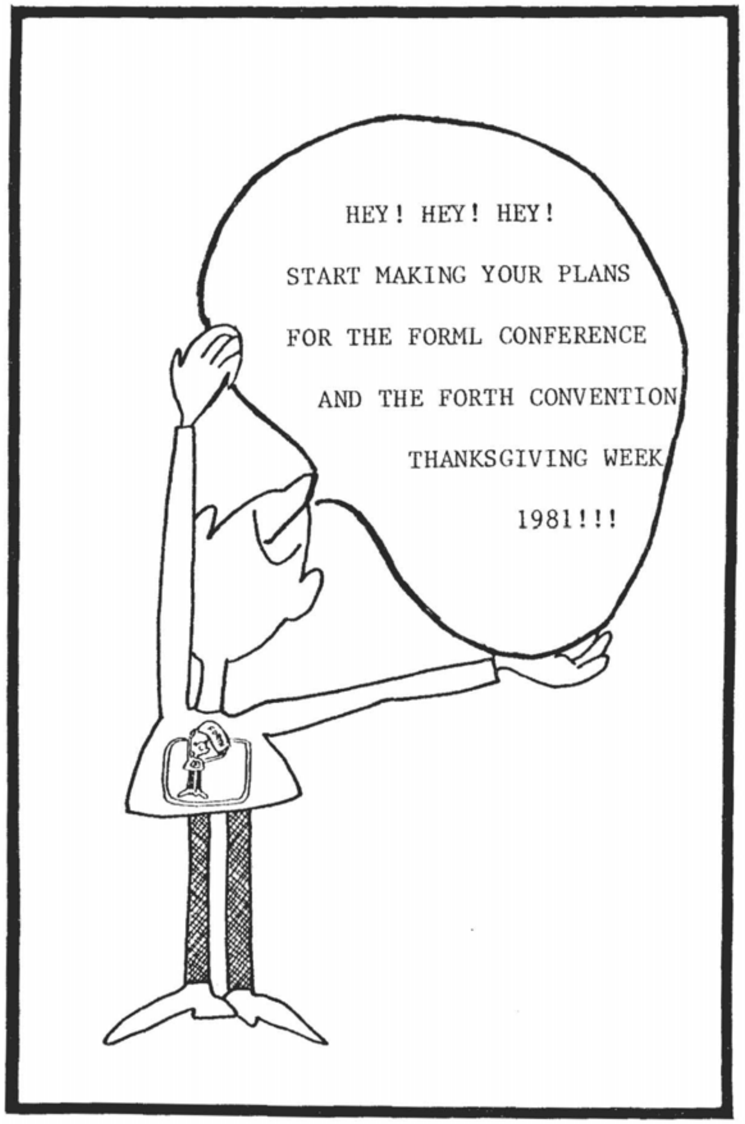
VISA #________________MASTER CHARGE #________________

Expiration Date______________ (Minimum of $10.00 on charge cards)

Make check or money order in US Funds on US bank, payable to: **FIG**. All prices include postage. ***No purchase orders without check.***

ORDER PHONE NUMBER: (415) 962-8653

FORTH INTEREST GROUP   PO BOX 1105   SAN CARLOS, CA 94070



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

FORTH INTEREST GROUP
P.O. Box 1105
San Carlos, CA 94070

TIME DATED MATERIAL
DELIVER BEFORE JULY 31

Address Correction Requested